(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENTAMT

(12) **Offenlegungsschrift**

(10) **DE 42 14 184 A 1**

(51) Int. Cl.[5]: G 06 F 12/00

(21) Aktenzeichen: P 42 14 184.2
(22) Anmeldetag: 30. 4. 92
(43) Offenlegungstag: 12. 11. 92

DE 42 14 184 A 1

(30) Unionspriorität: (32) (33) (31)

06.05.91 US 695952

(71) Anmelder:

Intel Corp., Santa Clara, Calif., US

(74) Vertreter:

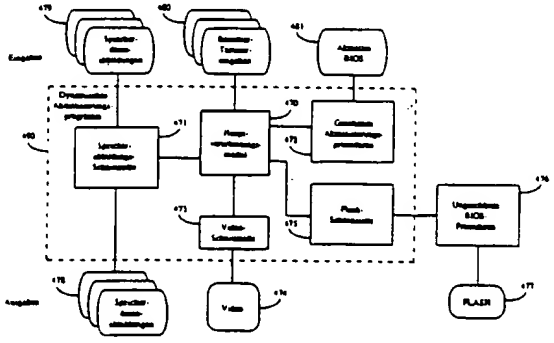
Zenz, J., Dipl.-Ing., 4300 Essen; Helber, F., Dipl.-Ing., 6144 Zwingenberg; Hosbach, H., Dipl.-Ing., 4300 Essen; Läufer, M., Dipl.-Chem. Dr.rer.nat., Pat.-Anwälte, 6144 Zwingenberg

(72) Erfinder:

Christeson, Orville H., Portland, Oreg., US; Gabel, Douglas L., Aloha, Oreg., US; Murphy, Sean T., Portland, Oreg., US

(54) Computersystem mit einem nicht-flüchtigen Speicher und Verfahren zu dessen Aktualisierung

(57) In dem Computersystem kann ein Teil des Befehlscodes/der Daten, die in einem nicht-flüchtigen Speicherbauelement gespeichert sind, dynamisch geändert oder aktualisiert werden, ohne irgendwelche Verkleidungen oder Teile vom Computersystem zu entfernen. Der Flash-Speicher zum Speichern nicht-flüchtiger Daten ist mit einem Systembus und weist vier nicht symmetrische, separat lösch- und programmierbare Speicherblöcke auf. Einer dieser vier Blöcke kann elektronisch gesperrt werden, um ein Löschen oder Ändern seines Inhalts zu vermeiden. Diese Konfiguration gestattet der Verarbeitungslogik des Computersystems, einen ausgewählten Block zu ändern oder zu aktualisieren, ohne den Inhalt eines anderen Blocks zu beeinflussen. Ein Speicherblock enthält ein normales BIOS. Ein elektronisch gesperrter Flash-Speicherbereich speichert ein Wiederherstellungs-BIOS. Das Computersystem weist außerdem Hardware-Mittel zum Einstellen entweder eines normalen oder eines Wiederherstellungs-Aktualisierungsmodus auf. Mit Hilfe eines Modusauswahlmittels wird entweder ein normales System-BIOS oder ein Wiederherstellungs-BIOS aktiviert.



DE 42 14 184 A 1

Beschreibung

Die Erfindung betrifft ein Computersystem. Insbesondere bezieht sich die Erfindung auf eine Computersystemarchitektur und einer nicht-flüchtigen Art der Verarbeitungslogik eines Grundbetriebssystems.

Viele bekannte Computersysteme sind üblicherweise zumindest mit einem Prozessor, einem Speicher mit wahlfreiem Zugriff (RAM) und einem Nur-Lese-Speicher (ROM) ausgestattet. Einige Systeme, beispielsweise verschiedene Taschenrechner, arbeiten nur mit einem Prozessor und einem ROM. ROM-Bauelemente stellen eine nicht-flüchtige Speicherart zur Verfügung, bei der die gespeicherten Informationen nicht zerstört werden, wenn das Computersystem von der Stromversorgung entfernt wird.

Bekannte Computersysteme sind typischerweise mit einem Anfangslader versehen (d. h. durch Einschalten zu initialisieren), wobei die im Inneren des Computersystems im ROM gespeicherte Verarbeitungslogik (d. h. Firmware) benutzt wird. Da der ROM nicht flüchtig ist, ist sichergestellt, daß die Firmware im ROM gültige Daten oder Befehle enthält; folglich können bekannte Computersysteme unter Benutzung der Firmware im ROM zuverlässig anfangsgeladen werden. Viele Computer haben erfolgreich diese Technik benutzt. Eines dieser Systeme ist der durch die IBM Corporation (in Armonk, N.Y., entwickelte IBM PC. Bekannte Versionen des IBM PC benutzen ROM-Bauelemente zur Speicherung der Firmware oder eines Basis-Eingabe-Ausgabe-Steuersystems (BIOS). Das BIOS stellt die Verarbeitungslogik des Betriebssystems dar, die die unterste Ebene der Softwaresteuerung der Hardware und der Ressourcen des Computersystems zur Verfügung stellt. Die ROM-Speicherung kann außerdem für eine nicht-flüchtige Ablage von Netzwerkkonfigurationsdaten oder anwendungsspezifischen Daten benutzt werden. Bekannte ROM-Bauelemente schließen die Grundform des Nur-Lese-Speichers (ROM), programmierbare Nur-Lese- Speicher (PROM) und löschbare programmierbare Nur-Lese-Speicher (EPROM) ein.

Obwohl ROM-basierte Computersysteme bisher sehr erfolgreich waren, gibt es eine Anzahl von Problemen bei der Benutzung dieser Bauelemente in einem Computersystem. Nur-Lese-Speicher müssen vor ihrem Einbau in das System während der Herstellung und Montage des Computersystems mit einem BIOS und/oder Daten programmiert werden. Häufig ist der BIOS-ROM auf einer Systemplatine innerhalb des Computergehäuses installiert. Um die Firmware eines ROM-basierten Computersystems zu ersetzen, zu modifizieren oder zu aktualisieren, müssen das Computergehäuse entfernt und die ROM-Bauelemente auf einer Systemplatine innerhalb des Computersystems herausgelöst und ersetzt oder umprogrammiert werden. Dieses das Bauelement angreifende Verfahren des Ersetzens und Umprogrammierens des ROMs ist aus mehreren Gründen von Nachteil. Erstens muß das Ersetzen des ROM im allgemeinen manuell durch qualifiziertes Kundendienst- oder Computerreparaturpersonal ausgeführt werden; folglich neigt der Vorgang dazu, aufwendig und zeitraubend zu sein. Zweitens können selbst bei qualifiziertem technischen Personal Probleme während des Ersetzens des ROMs auftreten. Wenn Lötverbindungen notwendig sind, können existierende Verbindungen während des Verfahrens zerstört oder erweicht werden. Auch können elektrostatische Entladungen unbeabsichtigte Zerstörungen bei anderen Komponenten auf der Leiterkarte während des Ersetzens des ROMs hervorrufen. Drittens können ROM-basierte Computersysteme nicht einfach auf spezielle Anwendungen zugeschnitten werden. Ein solches den Kundenwünschen entsprechendes Zuschneiden schließt Modifikationen für ein Arbeiten in nicht englischsprachigen Ländern ein. Um ein Computersystem durch Speicherung länderspezifischer Daten in einem nicht-flüchtigen Speicher auf Kundenwünsche zurechtzuschneiden, muß ein Benutzer ein Nur-Lese-Speicher auf einer Leiterplatte im Computersystem programmieren und installieren. Aufgrund der bisherigen unkomfortablen ROM-Installationsprozedur ist der Benutzer nicht oder kaum in der Lage, sein Computersystem auf seine Ansprüche zurechtzuschneiden.

Folglich besteht Bedarf an einer verbesserten Einrichtung zur Speicherung und Aktualisierung nicht-flüchtiger Befehlscodes und/oder Daten in einem Computersystem.

Die vorliegende Erfindung ist ein Computersystem, bei dem ein Teil der in einem nicht-flüchtigen Speicher gespeicherten Befehlscodes/Daten dynamisch geändert oder aktualisiert werden kann, ohne daß irgendeine Verkleidung oder Teile des Computersystems entfernt werden. Das Computersystem des bevorzugten Ausführungsbeispiels weist einen Bus zum Informationsaustausch, einen mit dem Bus gekoppelten Prozessor zur Informationsverarbeitung, einen mit dem Bus gekoppelten RAM zur Speicherung von Informationen und Befehlen für den Prozessor, ein mit dem Bus zum Austausch von Informationen und Befehlsaufrufen für den Prozessor gekoppeltes Eingabegerät, wie beispielsweise ein alphanumerisches Eingabegerät oder ein Cursorsteuergerät, ein mit dem Bus gekoppeltes Anzeigegerät zur Anzeige von Information zu einem Computerbenutzer und ein mit dem Bus gekoppeltes Datenspeichergerät, wie beispielsweise eine Magnetplatte oder ein Diskettenlaufwerk, zum Speichern von Informationen und Befehlen. Zusätzlich weist das Computersystem des bevorzugten Ausführungsbeispiels eine mit dem Bus gekoppelte Flash-Speicherbaugruppe zur Speicherung nicht-flüchtiger Befehlscodes und Daten auf. Zur Speicherung nicht-flüchtiger Befehlscodes und Daten können anstelle des Flash-Speichers auch andere Bauelemente benutzt werden. Bei Benutzung der Erfindung können die Inhalte des Flash-Speichers ohne eine Notwendigkeit zum Entfernen und/oder Ersetzen irgendeiner Hardwarekomponente des Computersystems ersetzt, geändert, aktualisiert oder umprogrammiert werden.

Der Flash-Speicher des bevorzugten Ausfuhrungsbeispiels enthält vier getrennt lösch- und programmierbare nicht symmetrische Speicherblöcke. Einer dieser vier Blöcke kann nach der Installation elektronisch gesperrt werden, um ein Löschen oder eine Änderung seines Inhalts zu verhindern. Diese Konfiguration gestattet der Verarbeitungslogik des Computersystems die Aktualisierung oder Änderung irgendeines ausgewählten Speicherblocks ohne eine Beeinflussung der Inhalte der anderen Blöcke. Ein Speicherblock enthält ein normales BIOS. Das BIOS enthält Befehle der Verarbeitungslogik, die durch den Prozessor ausgeführt werden. Ein zusätzlicher BIOS-Bereich kann benutzt werden, um den Speicherbereich des System-BIOS zu erweitern. Ein elektronisch geschützter (d. h. gesperrter) Bereich des Flash-Speichers wird zur Speicherung eines Wiederher-

stellungs-BIOS benutzt, welches für Wiederherstellungs-Operationen benutzt wird. Jeder dieser getrennt programmierbaren Bereiche des Flash-Speichers kann mit Hilfe des dynamischen Aktualisiermechanismus der vorliegenden Erfindung geändert oder aktualisiert werden.

Um zu vermeiden, daß eine abgebrochene BIOS-Aktualisierung das Computersystem funktionsunfähig macht, arbeitet das Aktualisierungsverfahren nach der Erfindung in zwei getrennten Benutzerumgebungen: einem normalen Aktualisierungsmodus und einem Wiederherstellungs-Aktualisierungsmodus. Im normalen Aktualisierungsmodus sind die Tastatur und die Videoausgabe des Computersystems zum Empfang von Befehlsaufrufen und zur Anzeige von Ergebnissen verfügbar. Der normale Aktualisierungsmodus schafft außerdem die Funktionalität zum Sichern, Überprüfen oder Aktualisieren der neben dem BIOS-Bereich zusätzlichen Bereiche des Flash-Speichers. Im Wiederherstellungs-Aktualisierungsmodus kann nur der Bereich des System-BIOS aktualisiert werden. Der Wiederherstellungs-Aktualisierungsmodus wird benutzt, wenn ein Benutzer das System nicht laden (booten) kann, weil das normale System-BIOS entweder infolge eines Stromversorgungsfehlers während einer normalen BIOS-Aktualisierung oder aus anderen Gründen verfälscht wurde.

Ein dynamisches BIOS-Aktualisierungsprogramm enthält den größten Teil der Verarbeitungslogik der Erfindung. Der verbleibende Teil der Verarbeitungslogik der Erfindung ist in der BIOS-Abbildung selbst enthalten. Die Verarbeitungslogik zur dynamischen Aktualisierung schließt die Logik zur Handhabung eines normalen Aktualisierungsmodus und des Wiederherstellungs-Aktualisierungsmodus ein.

Die Erfindung schließt außerdem Hardwaremittel zur Auswahl einer dieser zwei verfügbaren Aktualisierungsmoden ein. Im bevorzugten Ausführungsbeispiel sind diese Mittel zur Auswahl eines Aktualisierungsmodus als ein Jumper auf einer Leiterplatte im Computersystem implementiert.

Ist der Wiederherstellungsmodus mit Hilfe des Jumpers gesetzt, so kann der Prozessor des Computersystems durch Einschalten oder Rücksetzen initialisiert werden. Beim Einschalten oder Rücksetzen springt der Prozessor an eine Stelle innerhalb des geschützten Wiederherstellungs-BIOS-Blocks. Auf diese Weise kann der Flash-Speicher in einem Wiederherstellungs-Modus durch Aktivieren der sich im Wiederherstellungs-BIOS-Block aufhaltenden Verarbeitungslogik des Wiederherstellungsmodus umkonfiguriert werden. Folglich kann mit Hilfe der Modusauswahlmittel entweder ein normales System-BIOS oder ein Wiederherstellungs-BIOS aktiviert werden.

Bei Ausführung des normalen BIOS zeigt das dynamische Aktualisierungsprogramm ein Menü von Optionen zur Auswahl durch den Benutzer an. Diese Optionen schließen ein: die Überprüfung eines Flash-Speicherbereichs, das Sichern eines ausgewählten Flash-Speicherbereichs, das Aktualisieren eines Flashbereichs und das Verlassen des Programms. Bei Benutzung dieser Kommandooptionen können die aktualisierbaren Bereiche des Flash-Speichers den normalen System-BIOS-Bereich, einen für den Benutzer reservierten Bereich, einen Bereich für ein BIOS eines lokalen Netzwerks (LAN), einen SCSI-BIOS-Bereich, einen Videodatenbereich, andere Hardware- oder Software-spezifische BIOS- oder Daten-Bereiche oder einen Bereich des Flash-Speichers mit irgendeiner anderen anwendungsspezifischen Verarbeitungslogik einschließen.

Folglich liefert die Erfindung Mittel zur Speicherung nicht-flüchtiger Befehlscodes und Daten in einem Computersystem, wobei nicht-flüchtige Befehlscodes und Daten ohne Entfernen irgendeines Teils vom Computersystem aktualisiert werden können. Die Erfindung liefert ferner ein Mittel zum Aktualisieren eines ausgewählten Teils eines nicht-flüchtigen Speichers, während die anderen Teile unmodifiziert gelassen werden. Außerdem liefert die Erfindung ein Mittel zur Wiederherstellung im Falle eines Fehlers während der normalen Aktualisierungsprozedur des nicht-flüchtigen Speichers. Die Erfindung liefert auch Mittel zur Sicherung der aktuellen Inhalte des nicht-flüchtigen Speichers und zur Überprüfung des Inhalts der sich aktuell im nicht-flüchtigen Speicher aufhaltenden Befehlscodes und/oder Daten. Außerdem liefert die Erfindung Mittel zum Einbetten der Aktualisierungssteuerungssoftware in das System-BIOS, so daß die Aktualisierungssteuerungssoftware später ausgeblendet und zur Steuerung der Aktualisierungsprozedur des nicht-flüchtigen Speichers benutzt werden kann.

Im folgenden wird die Erfindung anhand eines in der Zeichnung dargestellten Ausführungsbeispiels näher erläutert. In der Zeichnung zeigen:

Fig. 1 eine Darstellung eines erfindungsgemäßen Computersystems;

Fig. 2 eine Darstellung der Speicherbelegung des BIOS in dem in der Erfindung benutzten Flash-Speicher;

Fig. 3a eine Speicherbelegung des BIOS während der normalen Arbeitsweise;

Fig. 3b eine Darstellung der Speicherbelegung des BIOS während des Wiederherstellungs-Modus;

Fig. 4 ein Blockdiagramm der Architektur der Verarbeitungslogik des dynamischen BIOS-Aktualisierungsmechanismus; und

Fig. 5 und 6 Flußdiagramme der erfindungsgemäßen Verarbeitungslogik der dynamischen BIOS-Aktualisierung.

Die vorliegende Erfindung stellt ein Computersystem dar, bei dem ein Teil des nicht-flüchtigen Speichers ohne ein Entfernen von Hardwarekomponenten des Systems geändert oder aktualisiert werden kann. In der folgenden Beschreibung werden zahlreiche spezielle Details aufgeführt, um ein besseres Verständnis der Erfindung zu erreichen. Für den Fachmann ist es jedoch klar, daß diese speziellen Details zur Realisierung der Erfindung nicht notwendig sind. An anderen Stellen werden bekannte Strukturen, Schaltungen und Schnittstellen nicht im Detail gezeigt, um das Verständnis der Erfindung nicht unnötig zu erschweren.

In **Fig.** 1 ist ein Blockdiagramm des erfindungsgemäßen Computersystems dargestellt. Das bevorzugte Ausführungsbeispiel der Erfindung wurde mit Hilfe eines durch den Anmelder der Erfindung hergestellten Mikroprozessors 80386 realisiert. Für den Fachmann ist es jedoch klar, daß auch andere Prozessoren und Computersystemarchitekturen benutzt werden können. Im allgemeinen weisen solche in **Fig.** 1 dargestellten Computersysteme einen Bus **100** zum Austauschen von Informationen, einen mit dem Bus gekoppelten Prozessor **101** zum Verarbeiten von Informationen, einen mit dem Bus **100** gekoppelten RAM **102** zum Speichern von Informationen und Befehlen für den Prozessor **101**, ein mit dem Bus **100** gekoppeltes Eingabegerät **104**, wie beispielsweise

ein alphanumerisches Eingabegerät oder ein Cursorsteuergerät, zum Austauschen von Informationen und Befehlsaufrufen mit dem Prozessor **101**, ein mit dem Bus **100** gekoppeltes Anzeigegerät **105** zum Anzeigen von Informationen für einen Computerbenutzer und ein mit dem Bus **100** gekoppeltes Datenspeichergerät, wie beispielsweise eine Magnetplatte und ein Diskettenlaufwerk, zur Speicherung von Informationen und Befehlen.

Zusätzlich weist das Computersystem des bevorzugten Ausführungsbeispiels eine mit dem Bus **100** gekoppelte Flash-Speicherbaugruppe **103** zum Speichern nicht-flüchtiger Befehlscodes und Daten auf. Die Flash-Speicherbaugruppe **103** stellt eine nichtflüchtige Speicherart zur Verfügung, die nicht gelöscht wird, wenn die Spannungsversorgung des Computersystems abgeschaltet wird; deren Speicherinhalte jedoch gelöscht und umprogrammiert werden können. Flash-Speicher sind hinreichend bekannt. Die Basis-Eingabe-Ausgabe-Verarbeitungslogik (BIOS) des Computersystems ist im Flash-Speicher **103** gespeichert. Zusätzlich können andere Systemsoftware- und anwendungsspezifische Parameter im Flash-Speicher **103** gespeichert sein. Beispielsweise kann ein Teil des Flash-Speichers zur Speicherung von Verarbeitungslogik für ein lokales Netzwerk (LAN) oder für ein SCSI-Interface benutzt werden. Die folgenden Abschnitte beschreiben, wie Teile des Flash-Speichers **103** ersetzt, geändert oder umprogrammiert (d. h. aktualisiert) werden ohne die Notwendigkeit zum Entfernen und/oder Ersetzen irgendeiner Hardwarekomponente des Computersystems.

Gegenwärtig sind verschiedene Typen nicht-flüchtiger Speicher bekannt, die ohne Entfernen des Bauelements von einer Leiterplatte, auf welcher das Bauelement installiert ist, umprogrammiert werden können. Eine Klasse umprogrammierbarer nicht-flüchtiger Speicher stellen die Flash-Speicher dar. Bekannt sind verschiedene Arten von Flash-Speichern. Mit Hilfe eines bestimmten Satzes elektrischer Signale können die Inhalte von Flash-Speichern gelöscht und mit neuen Daten umprogrammiert werden. Viele bekannte Flash-Speicher gestatten nur ein komplettes Löschen und Umprogrammieren aller Speicherplätze des Bauelements. Andere Flash-Speicher sind jedoch eingeteilt in getrennt löschbare und programmierbare Speicherblöcke in einem einzelnen Flash-Speicher. Im bevorzugten Ausführungsbeispiel der Erfindung wird ein solcher partitionierter Flash-Speicher benutzt. Es wird ein Flash-Speicher mit der Bezeichnung 28F001BT benutzt, der durch den Anmelder der vorliegenden Erfindung hergestellt wird. Für den Fachmann ist es klar, daß für die Erfindung andere Arten von umprogrammierbaren nicht-flüchtigen Speichern benutzt werden können. Eine Alternative zum Flash-Bauelement ist ein elektrisch löschbarer programmierbarer Nur-Lese-Speicher (EEPROM).

Der im bevorzugten Ausführungsbeispiel benutzte Flash-Speicher enthält vier getrennt lösch-/programmierbare nicht symmetrische Speicherblöcke. Einer dieser vier Blöcke kann elektronisch gesperrt werden, um ein Löschen oder Verändern des einmal installierten Inhalts zu verhindern. Diese Konfiguration gestattet es der Verarbeitungslogik des Computersystems, irgendeinen ausgewählten Speicherblock zu aktualisieren oder zu verändern, ohne die Inhalte der anderen Blöcke zu beeinflussen. In **Fig.** 2 sind verschiedene getrennt lösch-/programmierbare nicht symmetrische Blöcke des Flash-Speichers des bevorzugten Ausführungsbeispiels dargestellt. **Fig.** 2 zeigt eine normale BIOS-Speicherbelegung **200** der Inhalte des Flash-Speichers. Das BIOS enthält Befehle der Verarbeitungslogik, die durch den Prozessor ausgeführt werden. Im bevorzugten Ausführungsbeispiel der Erfindung ist der Prozessor des Computersystems ein Mikroprozessor 80386. Wenn die Stromversorgung des Computersystems des bevorzugten Ausführungsbeispiels eingeschaltet wird, springt der Prozessor zu einer Anfangsadresse FFFF0H und beginnt mit der Ausführung. Um die Initialisierung des Computersystems zu behandeln, muß folglich das aktive System-BIOS die Adresse FFFF0H einschließen. Wie durch die in **Fig.** 2 dargestellte normale BIOS-Speicherbelegung **200** gezeigt wird, springt der Prozessor nach dem Einschalten oder Rücksetzen des Prozessors zur einen Speicherplatz (einer Adresse) innerhalb eines normalen System-BIOS **201**. Die Verarbeitungslogik in dem Bereich **201** kann benutzt werden, um die normale Initialisierung und Steuerfunktionen des Computersystems zu behandeln. Der zusätzliche BIOS-Bereich **205** kann benutzt werden, um den Speicherbereich des System-BIOS zu erweitern.

Der elektronisch geschützte (d. h. gesperrte) Flash-Speicherbereich **202** ist ein Bereich zum Speichern eines für Wiederherstellungs-Operationen benutzten Wiederherstellungs-BIOS. Die Wiederherstellungs-Operation wird in weiter unten folgenden Abschnitten beschrieben. Der getrennt programmierbare Bereich **203** ist ein Speicherbereich, der für die Benutzung durch einen bestimmten Benutzer oder eine bestimmte Anwendung reserviert ist. Dieser Bereich kann benutzt werden, um die Arbeitsweise des Computersystems auf Kundenwünsche zuzuschneiden oder um die Funktionalität des System-BIOS zu erhöhen. Wie weiter unten beschrieben kann der getrennt programmierbare Bereich **203** LAN-Verarbeitungslogik, SCSI-Verarbeitungslogik, Videodaten oder Steuerlogik, andere Hardware- oder Software-spezifische Daten oder irgendeine andere anwendungsspezifische Verarbeitungslogik in einem Bereich des Flash-Speichers enthalten. Wenn der getrennt programmierbare Bereich **203** LAN- oder SCSI-Logik enthält, wird das System-BIOS zur Abarbeitung dieser zusätzlichen Flash-Bereiche aktiviert, um das Computersystem von einem Netzwerk oder einem anderen externen Gerät aus zu laden. Der getrennt programmierbare Bereich **204** ist ein für eine Benutzung durch das System reservierter Bereich. Dieser Bereich kann ein zusätzlicher Überlaufbereich eines normalen System-BIOS sein. Jeder dieser separat programmierbaren Bereiche des Flash-Speichers kann mit Hilfe des unten beschriebenen dynamischen Aktualisierungsmechanismus der Erfindung verändert oder aktualisiert werden.

Ein Nachteil bei der Benutzung eines Flash-Speichers zum Speichern eines BIOS ist die Möglichkeit eines Stromversorgungsausfalls oder eines anderen Fehlers während des Aktualisierungsprozesses. Das sich ergebende unvollständige oder zerstörte BIOS würde das System funktionsunfähig machen. Um die Arbeitsfähigkeit des Computersystems nach einer abgebrochenen BIOS-Aktualisierung wiederherzustellen, müßten Teile des Computersystems entfernt und umprogrammiert werden. Das Entfernen von Flash-Speicherbaugruppen kann zusätzlich dadurch kompliziert werden, daß oberflächenmontierte Flash-Bauelemente benutzt werden.

Um zu verhindern, daß eine abgebrochene BIOS-Aktualisierung das Computersystem funktionsunfähig macht, arbeitet das erfindungsgemäße Aktualisierungsverfahren in zwei getrennten Benutzerumgebungen: einem normalen Aktualisierungsmodus und einem Wiederherstellungs-Aktualisierungsmodus. Der Hauptunter-

schied zwischen diesen beiden Umgebungen ist die Benutzerschnittstelle. Im normalen Aktualisierungsmodus sind die Tastatur- und Videodienste des Computersystems für den Empfang von Befehlsaufrufen und die Anzeige von Ergebnissen verfügbar. Bei Benutzen der robusten Benutzerschnittstelle der normalen Aktualisierung kann der Benutzer eine Sicherungs-, Überprüfungs- oder Aktualisierungs-Operation für den Flash-Speicher im einzelnen angeben. Der Benutzer kann auch die Dateien im einzelnen angeben, die von verschiedenen Quellen, wie beispielsweise einem Floppy-Disk-Laufwerk, einem Festplattenlaufwerk, einem Netzwerk oder einem Modem wiedergewonnen oder gesichert werden sollen. Die in einer wiedergewonnenen oder gesicherten Datei enthaltenen Daten können eine BIOS-Abbildung, Datenabbildung oder eine andere Speicherbildrepräsentation des Inhalts eines nicht-flüchtigen oder Flash-Speicherbereichs ein. Der normale Aktualisierungsmodus schafft außerdem die Funktionalität zum Sichern, Überprüfen oder Aktualisieren anderer Bereiche des Flash-Speichers zusätzlich zu den BIOS-Bereichen.

Im Wiederherstellungs-Aktualisierungsmodus können nur die Bereiche des System-BIOS aktualisiert werden. Der Wiederherstellungs-Aktualisierungsmodus wird benutzt, wenn ein Benutzer das System nicht laden (booten) kann, weil das normale System-BIOS entweder infolge eines Stromausfalls während einer normalen BIOS-Aktualisierung oder aus irgendeinem anderen Grund zerstört wurde. In dieser Situation wird ein sich in dem geschützten Wiederherstellungs-BIOS-Block **202** (**Fig. 2**) aufhaltender getrennter Satz von Verarbeitungslogik allein für den Zweck der Aktualisierung des zerstörten normalen System-BIOS-Bereichs **201** ausgeführt. Um die durch die Verarbeitungslogik des Wiederherstellungs-BIOS belegte Speicherfläche zu reduzieren, wird nur ein minimaler Grad der Funktionalität zur Verfügung gestellt und keine Benutzerschnittstellen-Fähigkeit unterstützt. Die Tastatur- und Video-Dienste sind nicht verfügbar und es müssen vorgegebene Aktionen durch das Wiederherstellungs-BIOS ohne Benutzerinteraktionen ausgeführt werden. Von einem Lautsprecher des Computersystem ausgegebene hörbare Piepcodes zeigen den Status der Programmierung des BIOS-Bereichs an.

Bei der normalen Arbeitsweise des Computersystems des bevorzugten Ausführungsbeispiels führt der Prozessor **101** anfänglich Befehle außerhalb des normalen System-BIOS-Bereichs **201** des Flash-Speichers **103** aus. Nachfolgend kann ein vollständig ausgestattetes Betriebssystem vom Datenspeichergerät **106** in den RAM **102** eingelesen und ausgehend vom RAM **102** ausgeführt werden. Der Befehlscode des System-BIOS **201** kann außerdem für die Initialisierung und den normalen Zugriff zum Eingabegerät **104**, zum Anzeigegerät **105** und zum Datenspeichergerät **106** benutzt werden. Sobald das vollständig ausgestattete Betriebssystem zum RAM **102** übertragen wurde und in ihm abläuft, kann auf andere ausführbare Anwendungsdateien und Datendateien im Datenspeichergerät **106** zugegriffen werden. Eine dieser ausführbaren Dateien ist das dynamische Aktualisierungsprogramm für den nicht-flüchtigen Speicher, das den größten Teil der Verarbeitungslogik für die dynamische Aktualisierung gemäß der Erfindung enthält. Die verbleibenden Teile der erfindungsgemäßen Verarbeitungslogik halten sich in der Speicherabbildung selbst auf. Ebenfalls im Datenspeichergerät **106** halten sich Speicherabbildungen enthaltende Dateien und eine Datei, die eine für Aktualisierungen im Wiederherstellungsmodus benutzte vorgegebene Wiederherstellungs-BIOS-Abbildung enthält, auf. Eine Speicherabbildung, von welcher eine BIOS-Abbildung ein spezieller Typ ist, sind die binären Inhalte des Zielbereichs der Aktualisierung in einen nicht-flüchtigen oder Flash-Speicher. Die Verarbeitungslogik zur dynamischen Aktualisierung kann durch die für einen Fachmann bekannten Mittel aktiviert und ausgeführt werden. Die so aktivierte Verarbeitungslogik für eine dynamische Aktualisierung schließt die Logik zur Behandlung eines normalen Aktualisierungsmodus und des Wiederherstellungs-Aktualisierungsmodus ein. Die Architektur und die Arbeitsweise der Verarbeitungslogik zur dynamischen Aktualisierung wird im folgenden beschrieben.

Wie bereits gesagt arbeitet das bevorzugte Ausführungsbeispiel in zwei Grundmoden: einem normalen Aktualisierungsmodus und einem Wiederherstellungs-Aktualisierungsmodus. Die Erfindung schließt auch die Hardwaremittel zur Auswahl eines dieser beiden verfügbaren Aktualisierungsmoden ein. Im bevorzugten Ausführungsbeispiel sind diese Mittel zur Auswahl eines Aktualisierungsmodus als ein Jumper auf einer Leiterkarte im Computersystem realisiert. Der Jumper wird zur Änderung der Adreßleitung **16** in einer Schnittstelle zum Flash-Speicher benutzt. Durch Setzen des Jumpers in eine erste Einstellung können die normale in **Fig. 2** dargestellte BIOS-Speicherbelegung und das entsprechende normale BIOS-Aktualisierungshilfsprogramm benutzt werden. Wenn der Jumper in eine zweite Einstellung gesetzt wurde, ist die Wiederherstellungs-BIOS-Speicherbelegung konfiguriert und der Wiederherstellungs-Aktualisierungsmodus freigegeben. Für den Fachmann ist es offensichtlich, daß auch andere Mittel zur Auswahl eines dieser beiden Moden in einem Computersystem benutzt werden können, wie beispielsweise ein Schalter oder eine Drucktaste.

In den **Fig. 3a** und 3b ist die Wirkung des Jumpers beim Setzen des Wiederherstellungsmodus auf die Speicherkonfiguration des Flash-Speichers dargestellt. In **Fig. 3a** veranschaulicht eine normale BIOS-Speicherbelegung **200** die Konfiguration des Flash-Speichers, wenn der Jumper in der Stellung des normalen Modus gesetzt ist. Diese Konfiguration entspricht der oben beschriebenen und in **Fig. 2** dargestellten. Beim Einschalten oder nach einem System-Rücksetzen springt der Prozessor an eine Stelle im normalen System-BIOS **201**, die durch den Pfeil **311** in **Fig. 3a** angezeigt ist. Die Verarbeitungslogik im normalen System-BIOS **201** übernimmt dann die Steuerung und initialisiert das Computersystem für die normale Arbeitsweise. Wenn jedoch das normale System-BIOS **201** beispielsweise infolge einer abgebrochenen BIOS-Aktualisierung zerstört wurde, wird die Ausführung innerhalb des normalen BIOS **201** unvorhersehbar. Folglich wäre ohne die Möglichkeit eines Wiederherstellungsmodus das Computersystem mit einem zerstörten normalen System-BIOS **201** funktionsunfähig.

Wenn die Initialisierung des normalen System-BIOS **201** aufgrund eines zerstörten BIOS **201** erfolglos ist, kann der Wiederherstellungsmodus durch Umschalten der Auswahlmittel (d. h. des Jumpers) ausgewählt werden. Im Wiederherstellungsmodus ist die Adreßleitung **16**, die zur Adressierung von Speicherorten im Flash-Speicher benutzt wird, in einen komplementären Zustand geändert. Durch diese Änderung sind die untere und die obere Hälfte des Flash-Speichers logisch ausgetauscht. Folglich wird die Adresse FFFFFH auf den Speicher-

platz EFFFFH und die Adresse F0000H auf den Speicher E0000H abgebildet. Die umbelegte Speicherkonfiguration im Wiederherstellungsmodus ist in der in **Fig.** 3b dargestellten Wiederherstellungs-BIOS-Speicherbelegung dargestellt. Im Wiederherstellungsmodus ist das normale System-BIOS **201** auf den Speicherbereich **301** der Wiederherstellungs-BIOS-Speicherbelegung umgelegt. In ähnlicher Weise ist der Wiederherstellungs-BIOS-Block **202** auf den Ort **302** der Wiederherstellungs-BIOS-Speicherbelegung **300** umgelegt. Die Speicherbereiche **203**, **204** und **205** sind ebenfalls auf die Plätze **303**, **304** bzw. **305** umgelegt.

Sobald der Wiederherstellungsmodus mit Hilfe des Jumpers gesetzt ist und eine Wiederherstellungs-BIOS-Speicherbelegung wie in **Fig.** 3b dargestellt erreicht wurde, kann der Prozessor des Computersystems durch Einschalten oder Rücksetzen initialisiert werden. Beim Einschalten oder Rücksetzen springt der Prozessor zu einem Ort in dem geschützten Wiederherstellungs-BIOS-Block **302**, der durch den Pfeil **312** gezeigt ist. Auf diese Weise kann die Speicherbelegung des Flash-Bauelements in einem Wiederherstellungsmodus rekonfiguriert werden, wobei der Wiederherstellungsmodus die sich im Wiederherstellungs-BIOS-Block **302** aufhaltende Verarbeitungslogik aktiviert. Folglich kann mit Hilfe der Modusauswahlmittel entweder ein normales System-BIOS **201** oder ein Wiederherstellungs-BIOS **302** aktiviert werden. Sobald entweder das normale BIOS oder das Wiederherstellungs-BIOS aktiviert ist, kann die erfindungsgemäße dynamische Verarbeitungslogik zum Aktualisieren des BIOS aus dem Datenspeichergerät **106** wiedergewonnen und in den RAM **102** zur Ausführung durch den Prozessor **101** geladen werden. Für die dynamische Verarbeitungslogik zur Aktualisierung wird außerdem ein den Modus anzeigendes Datenfeld gesetzt. Diese Modusanzeige gibt im einzelnen an, ob das Computersystem in einem normalen oder einem Wiederherstellungsmodus arbeitet.

Sobald die erfindungsgemäße dynamische Aktualisierungs-Verarbeitungslogik aktiviert ist, arbeitet sie in einer unten näher beschriebenen und in den Flußdiagrammen der **Fig.** 5 und 6 dargestellten Art und Weise. Die dynamische Verarbeitungslogik zur Aktualisierung des BIOS stellt in Abhängigkeit davon, ob entweder ein normaler Modus oder ein Wiederherstellungs-Modus ausgewählt wurde, zwei Sätze von Merkmalen zur Verfügung. Im normalen Modus wird eine robuste Benutzerschnittstelle zur Verfügung gestellt. Diese Schnittstelle schafft grafische Repräsentationen für Auswahlmenüs, Dateiselektionen, Hilfsinformationen und Statusnachrichten zum Aktualisieren irgendeines Flash-Speicherbereichs einschließlich des System-BIOS-Speicherbereichs. Dieses Paket der grafischen Benutzerschnittstelle stellt Subroutinen zum Definieren und Anzeigen von farbigen Bildschirmanzeigen der Benutzerschnittstelle zur Verfügung. Dieses Paket unterstützt außerdem eine On-Line-Hilfe zur Unterstützung des Benutzers.

Im Wiederherstellungsmodus ist die Verarbeitungslogik zur Wiederherstellung des BIOS nur dazu bestimmt, die kritischen Abschnitte des Systems aufzubauen ohne den Vorteil der Konfigurationsinformation. Das Wiederherstellungs-BIOS nimmt an, daß das normale System-BIOS zerstört wurde. Folglich ist die einzige Funktion des Wiederherstellungs-BIOS, das System an einem Punkt freizugeben, wo das normale System-BIOS in das Flash-Speicher geladen werden kann. Das Wiederherstellungs-BIOS wird nach dem Einschalten automatisch ohne die Möglichkeit eines Benutzereingriffs ausgeführt. Von einem Lautsprecher des Computersystems aus gegebene hörbare Piepcodes zeigen den Status der Programmierung des BIOS-Bereichs an. Das Wiederherstellungs-BIOS ist ein Subset des normalen BIOS.

Wenn das dynamische Aktualisierungsprogramm außerhalb des normalen BIOS ausgeführt wird, zeigt es ein Menü von Optionen zur Auswahl durch den Benutzer an. Diese Optionen schließen ein: die Überprüfung eines Flash-Speicherbereichs, die Sicherung eines ausgewählten Flash-Speicherbereichs, die Aktualisierung eines Flash-Speicherbereichs und das Beenden. Dann korrespondiert das dynamische Aktualisierungsprogramm mit dem normalen System-BIOS, um die Bereiche des Flash-Speichers zu bestimmen, die für ein Lesen oder Schreiben verfügbar sind. Dann wird ein Submenü angezeigt, das die zum Lesen oder Schreiben verfügbaren Speicherbereiche anzeigt. Der Benutzer kann dann den Bereich von diesem Menü für eine Überprüfungs-, Sicherungs- oder Aktualisierungs-Operationen auswählen. Zusätzliche Menüs werden angezeigt, um den Benutzer zu veranlassen, Dateinamen für Sicherungs-/Überprüfungs-/Aktualisierungs-Operationen und für eine Überprüfung von Dateiinformationen vor einer Aktualisierung einzugeben. Außerdem werden zusätzliche Menüs benutzt, um dem Benutzer zusätzliche Informationen über eine gesicherte Dateiabbildung zur Verfügung zu stellen. Die Operation wird dann ausgeführt und der Benutzer wird über den Status der Fertigstellung der Operation informiert. Ein normaler Aktualisierungsmodus kann zur Aktualisierung entweder des normalen System-BIOS-Bereichs **201** und **205**, des für den Benutzer reservierten Bereichs **203** oder des für das System reservierten Bereichs **204** benutzt werden. Diese Bereiche können entweder getrennt oder kombiniert aktualisiert werden. Der getrennt programmierbare Bereich **203** oder andere getrennt programmierbare Bereiche können LAN-Verarbeitungslogik, SCSI-Verarbeitungslogik, andere Netzwerklogik oder Daten, Videodaten oder Steuerlogik, andere hardware- oder softwarespezifische Logik oder Daten oder irgendeine andere anwendungsspezifische Verarbeitungslogik in einem Bereich des Flash-Speichers enthalten. Die Bereiche **201** und **205** sind in demselben physikalischen Block enthalten und werden folglich gleichzeitig gelöscht.

Wie bereits oben erwähnt kann das dynamische Aktualisierungsprogramm zur Aktualisierung irgendeines Bereichs des Flash-Speichers benutzt werden. Die Aktualisierung des normalen System-BIOS-Bereichs wurde oben beschrieben. Erweiterungen der Erfindung schließen die Aktualisierung des Bereichs **203** oder anderer getrennt programmierbarer Bereiche mit LAN-Verarbeitungslogik, SCSI-Verarbeitungslogik, Videodaten oder Steuerlogik, anderer hardware- oder softwarespezifischer Logik oder Daten oder irgendeiner anderen anwendungsspezifischen Verarbeitungslogik in einem Bereich des Flash-Speichers ein. Wenn ein LAN-Verarbeitungslogik, SCSI-Verarbeitungslogik oder Netzwerklogik oder -daten enthaltender Flash-Speicherbereich aktualisiert wird, kann das Computersystem für eine bestimmte Netzwerkverbindung konfiguriert werden. In bekannten vernetzten Computersystemen ist ein Netzwerk-Lade-PROM für jedes spezielle Netzwerk notwendig. Es ist notwendig, die PROMs auszutauschen oder eine andere Netzwerkleiterkarte hinzuzufügen, um ein Computersystem für ein bestimmtes Netzwerk umzukonfigurieren.

Durch Benutzung der Erfindung können andere Flash-Speicherbereiche neben dem System-BIOS-Bereich, wie beispielsweise der Bereich **203**, mit LAN-Verarbeitungslogik, SCSI-Verarbeitungslogik, anderer Netzwerklogik oder -daten, Videodaten oder Steuerlogik, anderer hardware- oder softwarespezifischer Logik oder Daten oder irgendeiner anderen anwendungsspezifischen Verarbeitungslogik in einem Bereich des Flash-Speichers programmiert werden. Die Programmierung dieser nicht zur System-BIOS-Aktualisierung gehörigen Bereiche wird mit Hilfe der gleichen dynamischen Aktualisierungsverfahren ausgeführt, die für die Aktualisierung des System-BIOS-Bereichs beschrieben wurden. Zusätzlich kann das System-BIOS so konfiguriert werden, daß der nicht zur System-BIOS-Aktualisierung gehörige Bereich beim Laden des Systems abgetastet wird, um zu ermitteln, ob der sich dort aufhaltende Befehlscode ausgeführt werden soll, um das Computersystem in einer anderen Art und Weise, beispielsweise von einem Netzwerk aus, zu laden (zu booten). Folglich kann die Realisierung der dynamischen Aktualisierung eines nicht zum System-BIOS gehörigen Bereichs des Flash-Speichers einen zusätzlichen Schritt zum Freigeben des Abtastens des nicht zur System-BIOS-Aktualisierung gehörigen Bereiches durch das System-BIOS einschließen.

Dieser Freigabeschritt kann auf verschiedene Weise ausgeführt werden. Zunächst kann ein Hardwareschalter im Computersystem oder eine bestimmte Tastenkombination zum Aktivieren des dem Programmieren des Bereichs folgenden Abtastens des aktualisierten Bereichs vorgesehen sein. Ein alternatives Ausführungsbeispiel des Freigabeschrittes sieht ein Aktualisieren des System-BIOS-Bereichs nach einem Aktualisieren des nicht zum System-BIOS gehörigen Bereichs vor. Beim Aktualisieren des System-BIOS-Bereichs kann ein Datenfeld gesetzt werden, das die Abtastung des nicht zum System-BIOS gehörigen Bereichs freigibt. In einem dritten Ausführungsbeispiel kann das System-BIOS so konfiguriert sein, daß es stets den nicht zum System-BIOS gehörigen Bereich abtastet und hier den Befehlscode ausführt, wenn ein bestimmtes Datenmuster an einem festen Speicherplatz in dem nicht zum System-BIOS gehörigen Bereich erkannt wird. In jedem dieser Ausführungsbeispiele kann ein nicht zum System-BIOS gehöriger Bereich des Flash-Speichers aktualisiert werden, wobei die Notwendigkeit einer netzwerkspezifischen Hardware oder eines PROM-Bauelements beseitigt wird. Mit Hilfe des aktualisierten, nicht zum System-BIOS gehörigen Bereichs kann ein Computersystem von einem Netzwerk aus geladen (gebootet) werden oder in einer Konfiguration ohne Disketten arbeiten.

In **Fig. 4** ist die Architektur des dynamischen Aktualisierungsprogramms der Erfindung dargestellt. Im bevorzugten Ausführungsbeispiel gibt es fünf getrennte Kompiliermodule (**470**, **471**, **472**, **473**, **475**), die miteinander verbunden sind, um das dynamische Aktualisierungsprogramm **490** der Erfindung zu bilden. Der Hauptverarbeitungsmodul **470** übernimmt die gesamte Steuerung, Entscheidungsbildung und die Schnittstellen zu den verbleibenden vier Modulen. Der Speicherabbildungs-Schnittstellenmodul **471** wird benutzt, um Speicherabbildungen zu bearbeiten (d. h. zu lesen und zu schreiben). Eine Speicherabbildung ist der binäre Inhalt des Zielbereichs im nicht-flüchtigen Speicher. Speicherabbildungen werden von einem externen Speichermedium, wie beispielsweise einem magnetischen Plattenlaufwerk, gelesen, welches eine Vielzahl von Speicherabbildungsdateien **479** enthält. Speicherabbildungen werden in Dateien **478** in dem externen Speichermedium zum dauerhaften Verbleib geschrieben. Die Speicherabbildungsdateien **479** enthalten jeweils einen Kopfteil mit verschiedenen Informationen über die Speicherabbildung.

Um einen nicht-flüchtigen Speicherbereich zu aktualisieren, muß Aktualisierungslogik verwendet werden, die sich nicht in dem zu aktualisierenden nicht-flüchtigen Speicher aufhält. Wenn Logik benutzt würde, die sich in dem zu aktualisierenden nichtflüchtigen Speicher aufhält, könnte die Aktualisierungsoperation diese Logik ändern oder zerstören. Um das Ziel zu erreichen, keine hardwarespezifischen Prozeduren zu implementieren, ist es vorteilhaft, die notwendige Aktualisierungslogik in jeder der zu aktualisierenden Speicherabbildungen **479** zu speichern. Diese Aktualisierungslogik ist in den Speicherabbildungen **479** in Form geschützter Softwareprozeduren gespeichert. Ein wichtiger Grund für die Speicherung der geschützten Prozeduren in den Speicherabbildungen **479** selbst ist die Zuteilung der hardwareabhängigen Befehlscodes zu den Speicherabbildungen **479**, während die hardwareunabhängigen Befehlscodes in dem aus dem RAM heraus ausgeführten dynamischen Aktualisierungsprogramm **490** gehalten werden: Folglich kann das gleiche dynamische Aktualisierungsprogramm **490** für viele verschiedene Hardwarekonfigurationen benutzt werden, da die Aktualisierungslogik in jeder Speicherabbildung **479** die hardwarespezifischen Angelegenheiten handhabt. Wenn ein Benutzer eine Aktualisierungsoperation anfordert, fordert das dynamische Aktualisierungsprogramm **490** das normale System-BIOS **481** auf, diese Prozeduren in eine durch das dynamische Aktualisierungsprogramm **490** spezifizierte sichere RAM-Position zu speichern. Diese Arbeitsweise sichert, daß die Speicherabbildung selbst diese geschützten Prozeduren definiert und aufrechterhält und daß diese sich nicht in dem zu aktualisierenden Adreßbereich während der Aktualisierungen des Flash-Speichers aufhalten. Der Modul der geschützten Aktualisierungsprozeduren **472** ist dafür verantwortlich, diese geschützten Prozeduren von der Aktualisierungsspeicherabbildung **479** zu erhalten und sie dem Hauptverarbeitungsmodul **470** des dynamischen Aktualisierungsprogramm **490** zu Verfügung zu stellen.

Der Videoschnittstellenmodul **473** führt alle geforderten Interaktionen mit dem Videoanzeigegerät **474** aus. Diese Interaktionen schließen die Anzeige von Menüs, bestimmten Fehlernachrichten und Statusnachrichten und das Erlangen der geforderten Informationen über die Orte der Speicherabbildung (d. h. Dateinamen) ein. Benutzertastatur-Eingabemittel **480** sind ebenfalls zum Empfangen von Benutzereingaben und Befehlsaufrufen vorgesehen. Der Flash-Interfacemodul **475** stellt eine Brücke zwischen einem Satz ungeschützter BIOS-Prozeduren **476** und dem Hauptverarbeitungsmodul **470** zu Verfügung. Die ungeschützten BIOS-Prozeduren **476**, wie beispielweise "Lies den. Flash-Speicher" erfordern nicht die Benutzung der früher definierten geschützten Prozeduren. Auf diese ungeschützten BIOS-Prozeduren **476** wird über ein Prozessor-Interrupt (INT **15**) zugegriffen, wobei ein Zugriff auf das Flash-Speicher **477** zur Verfügung gestellt wird. Die Schnittstellenspezifikationen sowohl für die geschützten als auch für die ungeschützten Prozeduren werden am Ende der Beschreibung geliefert.

Arbeitsweise des bevorzugten Ausführungsbeispiels

Die Verarbeitungslogik des dynamischen Aktualisierungsprogramms des bevorzugten Ausführungsbeispiels ist betriebsfähig im RAM **102** angeordnet und wird durch den Prozessor **101** des oben beschriebenen Computersystems ausgeführt. Die erfindungsgemäße Verarbeitungslogik kann in äquivalenter Weise in anderen Speichermitteln, auf die der Prozessor **101** zur Ausführung zugreifen kann, angeordnet sein. Die Verarbeitungslogik des dynamischen Aktualisierungsprogramms **490** kann eine getrennt kompilierte und geladene Einheit sein oder als ein Teil in ein größeres Softwaresystem eingegliedert werden. In jedem Falle können Mittel zum Aktivieren der erfindungsgemäßen Verarbeitungslogik mit Hilfe der oben beschriebenen Techniken ausgeführt werden. Sobald sie aktiviert wurde, arbeitet die erfindungsgemäße Verarbeitungslogik in der unten beschriebenen und in den Flußdiagrammen der **Fig.** 5 und 6 dargestellten Weise.

Wie in **Fig.** 5 dargestellt, startet das erfindungsgemäße dynamische Aktualisierungsprogramm bei seiner Aktivierung am Block **501**. Am Entscheidungsblock **502** wird ein Test ausgeführt, um zu bestimmen, ob der normale oder der Wiederherstellungs-Modus ausgewählt wurde. Wie zuvor beschrieben setzen sowohl das normale BIOS als auch das Wiederherstellungs-BIOS ein den Modus anzeigendes Datenfeld, wenn eines der BIOS aktiviert wurde. Durch Zugriff auf dieses Datenfeld kann im Entscheidungsblock **502** der aktive Modus bestimmt werden. Wurde der Wiederherstellungs-Modus ausgewählt, wird der Verarbeitungspfad **503** zum Verarbeitungsblock **505** beschritten, wo eine vorgegebene normale System-BIOS-Abbildung aus dem Datenspeichergerät wiedergewonnen, d. h. in den RAM eingelesen wird. Als nächstes wird auf das Wiederherstellungs-BIOS zugegriffen, um die geschützten BIOS-Aktualisierungs-Prozeduren wiederzugewinnen (Verarbeitungsblock **506**). Die in Block **506** wiedergewonnenen (retrieved) geschützten Prozeduren schließen eine Prozedur zum Löschen des BIOS und eine Prozedur zum Programmieren des BIOS ein. Als nächstes wird das Computersystem auf die Löschung und Programmierung des Flash-Speichers vorbereitet (Verarbeitungsblock **535**). Die Vorbereitung des Computersystems auf das Löschen und die Programmierung des Flash-Speichers schließt Operationen zum Sperren der Cache- und Shadowing-Funktionen, zur Freigabe der Schreiboperationen zum Flash-Speicher und andere hardwarespezifische Operationen ein. Sobald die geschützten Prozeduren wiedergewonnen und das Computersystem vorbereitet wurden, wird die Prozedur zum Löschen des BIOS aktiviert, um den normalen System-BIOS-Bereich im Flash-Speicher zu löschen. Sobald die Lösch-Operation beendet wurde, wird die Prozedur zum Programmieren des BIOS benutzt, um die vordefinierte BIOS-Abbildung in den normalen System-BIOS-Bereich des Flash-Speichers zu laden (Programmierblock **507**). Wenn diese Operation abgeschlossen wurde, endet die Verarbeitungslogik des dynamischen Aktualisierungsprogramms am Verarbeitungsblock **508**. Der Benutzer wird durch einen vom Lautsprecher des Computersystems ausgegebenen hörbaren Piepcode über den Abschluß informiert. Nach Abschluß der Abarbeitung des Wiederherstellungs-Modus können der Jumper bzw. die Modusauswahlmittel zur Auswahlposition des normalen Modus zurückgeschaltet werden und das Computersystem kann erneut gestartet oder durch Einschalten initialisiert werden, um die Steuerung auf das frisch geladene normale System-BIOS im Flash-Speicher zu übertragen.

Wenn ein normaler Modus ausgewählt wurde, führt der Verarbeitungspfad **504** vom Entscheidungsblock **502** zum Verarbeitungsblock **510**, wo ein Hauptmenü von normalen Aktualisierungsoptionen dem Benutzer angezeigt wird. Diese Optionen schließen eine Operation zur Überprüfung eines Flash-Bereichs, eine Operation zum Sichern eines Flash-Bereichs, eine Operation zum Aktualisieren eines Flash-Bereichs und eine Beendigungsoperation ein. Sobald ein Menü dem Benutzer präsentiert wird, wird im Verarbeitungsblock **511** eine Menüauswahl vom Eingabegerät **104** gewonnen. Wenn die Beendigungsoperation ausgewählt wurde, führt vom Entscheidungsblock **512** der Verarbeitungspfad **513** zum Beendigungsblock **515**, wo die Ausführung des dynamischen Aktualisierungsprogramms abgebrochen wird.

Wenn jedoch die Beendigungsoperation nicht ausgewählt wurde, wird der Verarbeitungspfad **514** zum Verarbeitungsblock **516** genommen, wo eine ungeschützte normale System-BIOS-Prozedur aktiviert wird, um die aktiven Flash-Bereiche des BIOS, die überprüft, gesichert oder aktualisiert werden können, zu gewinnen. Die aktiven Flash-Bereiche werden im Verarbeitungsblock **517** in einem sekundären Menü angezeigt. Der Benutzer wird wiederum aufgefordert, im Verarbeitungsblock **518** eine Menüauswahl über das Eingabegerät **104** zu treffen. Wenn eine Überprüfung eines Flash-Bereichs ausgewählt wurde, führt der Verarbeitungspfad **520** zu dem in **Fig.** 6 dargestellten mit A bezeichneten Kreis. Wenn eine Operation zum Sichern eines Flash-Bereichs ausgewählt wurde, führt der Verarbeitungspfad **523** zu dem mit B bezeichneten Kreis in **Fig.** 6. Wenn eine Operation zum Aktualisieren eines Flash-Bereichs ausgewählt wurde, führt in ähnlicher Weise der Verarbeitungspfad **526** zu dem mit C bezeichneten Kreis in **Fig.** 6. Wenn eine ungeeignete (d. h. Rückkehr zum Hauptmenü) Befehlsauswahl getroffen wurde, führt der Verarbeitungspfad **527** zum mit D bezeichneten Kreis, wo die Steuerung zum Verarbeitungsblock **510** zurückkehrt, wo das Hautmenü erneut dem Benutzer präsentiert wird.

In **Fig.** 6 ist die Verarbeitungslogik für jede der drei Flash-Speicherbereichs-Operationen dargestellt. Wenn eine Operation zum Überprüfen des Flash-Speichers ausgewählt wurde, wird die Verarbeitungslogik unter dem mit A bezeichneten Kreis ausgeführt. Im Verarbeitungsblock **601** wird der Benutzer aufgefordert, den Namen einer Datei einzugeben, die mit dem spezifizierten Flash-Speicherbereich verglichen werden soll. Wenn eine einzelne Datei nicht groß genug ist, um den gesamten Inhalt des spezifizierten Flash-Speicherbereichs zu enthalten, können verschiedene Dateien, die Teile der Flash-Datenvergleichsabbildung enthalten, miteinander verkettet werden, wobei eine bekannte Dateiverkettungstechnik angewendet wird. In der Schleife zwischen dem Verarbeitungsblock **602** und dem Entscheidungsblock **604** werden die Inhalte der Vergleichsdatei oder der verketteten Dateien gelesen und mit den Inhalten des spezifizierten Flash-Speicherbereichs verglichen. Wenn Differenzen gefunden werden, wird die Schleife mit einem Überprüfungs-Fehler beendet. Das Überprüfungsverfahren wird fortgesetzt, bis der gesamte Inhalt der Vergleichsdatei und der zugeordneten verketteten Dateien mit den Inhalten des gewählten Flash-Speicherbereichs verglichen wurde. Wenn die Überprüfung vollständig ist,

führt der Verarbeitungspfad **606** zum Verarbeitungsblock **607**, wo die Vergleichsergebnisse dem Benutzer angezeigt werden. Die Verarbeitung wird dann an dem in **Fig. 5** dargestellten mit E bezeichneten Kreis fortgesetzt, wo der Bediener zur nächsten Befehlsauswahl aufgefordert wird.

In **Fig. 6** wiederum ist unter dem mit B bezeichneten Kreis die mit dem Kommando zum Sichern eines Flash-Speicherbereichs verbundene Verarbeitungslogik dargestellt. Wenn der Benutzer ein Kommando zum Sichern eines Flash-Speicherbereichs auswählt, wird die Verarbeitungssteuerung an den Verarbeitungsblock **608** übergeben, wo der Benutzer zur Eingabe eines Namens einer Datei aufgefordert wird, welche zur Speicherung des ausgewählten Flash-Speicherbereichs benutzt werden soll. Mehrere Dateien können mit Hilfe bekannter Techniken miteinander verkettet werden, wenn eine einzelne Datei nicht groß genug ist, um den gesamten ausgewählten Flash-Speicherbereich zu speichern. Anschließend wird eine Schleife zwischen dem Verarbeitungsblock **609** und dem Entscheidungsblock **611** initialisiert, wo die Inhalte des ausgewählten Flash-Speicherbereichs gelesen (Verarbeitungsblock **609**) und dann in die spezifizierte Datei oder die verketteten Dateien geschrieben (Verarbeitungsblock **610**) werden. Die Sicherungsoperation wird fortgesetzt, bis der gesamte Inhalt des ausgewählten Flash-Speicherbereichs in die spezifizierte Datei oder die Dateien übertragen wurde. Wenn dies der Fall ist, führt der Verarbeitungspfad **613** zum Verarbeitungsblock **614**, wo der Status der Sicherungsoperation dem Benutzer angezeigt wird. Die Verarbeitung wird dann an dem in **Fig. 5** dargestellten mit E bezeichneten Kreis fortgesetzt, wo der Bediener zur Eingabe des nächsten Befehlsaufrufs aufgefordert wird.

Wiederum in **Fig. 6** ist die Verarbeitungslogik des Kommandos zum Aktualisieren des Flash-Bereichs unter den mit C bezeichneten Kreis dargestellt. Wenn das Kommando zum Aktualisieren des Flash-Bereichs durch den Benutzer ausgewählt wurde, wird die Verarbeitungssteuerung an den Verarbeitungsblock **615** übertragen, wo der Benutzer zur Eingabe eines Namens einer Datei aufgefordert wird, die eine Abbildung der Flash-Daten enthält, die in den spezifizierten Flash-Speicherbereich übertragen werden sollen. Damit die Aktualisierungs-Utility der Erfindung verschiedene Computerkonfigurationen und Flash-Speicher handhaben kann, ist die Aktualisierungs-Verarbeitungslogik im normalen System-BIOS als geschützte Prozedur eingebettet. Das Computersystem wird für die Löschung und Programmierung des Flash-Speichers vorbereitet (Verarbeitungsblock **635**). Der Flash-Speicherbereich wird im Verarbeitungsblock **636** gelöscht. Diese geschützten Prozeduren werden aus dem BIOS im Verarbeitungsblock **616** zur Benutzung durch die erfindungsgemäße dynamische Aktualisierungs-Utility wiedergewonnen. Diese geschützten Aktualisierungsprozeduren werden durch die dynamische Aktualisierungs-Utility benutzt, um die Inhalte der spezifizierten Datei oder der Dateien in die spezifizierten Flash-Speicherbereiche zu übertragen (Verarbeitungsblock **617**). Die Aktualisierungsoperation wird fortgesetzt, bis der gesamte Inhalt der spezifizierten Datei zum spezifizierten Flash-Speicherbereich übertragen wurde. Wenn die Aktualisierung vollständig ist (Verarbeitungspfad **620**) werden die Status-Ergebnisse der Aktualisierungsoperationen dem Benutzer im Verarbeitungsblock **621** angezeigt. Wenn die Aktualisierung ein erneutes Laden des Computersystems erfordert (Verarbeitungspfad **623**), kann das gerade aktualisierte BIOS durch erneutes Anfangsladen des Computersystems im Verarbeitungsblock **625** aktiviert werden. Wenn kein erneutes Laden erforderlich ist, führt der Verarbeitungspfad **624** zu dem in **Fig. 5** dargestellten mit E bezeichneten Kreis, wo der Benutzer zur Eingabe einer neuen Kommandoauswahl aufgefordert wird.

Es wurde ein Computersystem beschrieben, bei dem ein Teil der in einem nicht-flüchtigen Speicher gespeicherten Befehlscodes/Daten dynamisch geändert oder aktualisiert werden kann, ohne Verkleidungen oder Teile vom Computersystem zu entfernen.

BIOS-PROZEDUREN DES FLASH-SPEICHERS

A. Ungeschützte BIOS-Prozeduren

Die folgenden BIOS-Serviceaufrufe sind für die Flash-Speicherschnittstelle.

INTERRUPT = 15H
FUNKTION (AH) = 0DBH
SUBFUNKTIONEN (AL)

00H Lesen einen Flash-Speicherbereich

Eingabe:
DS = Segment des Ausgabedateikopfteils
SI = Offset des Ausgabedateikopfteils
ES = Segment des Puffers zum Plazieren der gelesenen Informationen
DI = Offset des Puffers zum Plazieren der gelesenen Informationen

Ausgabe:
AH = Statusausgabe
0 = erfolgreiches Lesen
1 = ungültige Eingabe

01H Berichten über Flash-Speicherbereiche

Eingabe:
CL = Logische Bereichsnummer (0—0FFh)

Ausgabe:
DI = Offset des Zeigers auf den 32-Byte-Bereich der Info-Struktur
ES = Segment des Zeigers auf den 32-Byte-Bereich der Info-Struktur
AH = Statusausgabe
0 = erfolgreich
1 = ungültige Eingabe

02H Empfangen der Größe der geschützten Prozedur

Eingabe:
keine
Ausgabe:
BX = Größe der geschützten Prozeduren in Bytes

03H Vorbereiten zum Flash-Löschen

Eingabe:
ES = Segment des Puffers zum Plazieren der geschützten Prozeduren
DI = Offset des Puffers zum Plazieren der geschützten Prozeduren
DS = Segment des Kopfteils der Eingabedatei
SI = Offset des Kopfteils der Eingabedatei
BX = Segment des Puffers für BIOS-Tabellen
DX = Offset des Puffers für BIOS-Tabellen

Ausgabe:
BX = Menge des durch das BIOS geforderten zusätzlichen Speichers zum Abschließen der Operation
AH = Statusausgabe
0 = erfolgreicher Abschluß
1 = ungültige Eingabe
2 = ungültige Operation
3 = ungültige Größe
4 = ungültiger Datentyp

04H Ist der Wiederherstellungs-Modus aktiv?

Eingabe:
keine

Ausgabe:
BX =
0 = Wiederherstellungs-Modus aktiv
nicht 0 = normaler Modus aktiv
AH = Statusausgabe
0 = erfolgreicher Abschluß
1 = Flash-Speicher wird von diesem System nicht unterstützt

B. Geschützte BIOS-Prozeduren

Auf die geschützten Prozeduren wird zugegriffen durch einen weiten Aufruf (far call) zu den Adressen, die in ES : DI durch den "Vorbereiten zum Löschen"-Interrupt-Aufruf zurückgegeben wurden und sie erfordern die folgenden Schnittstellen:

AL = 00H Lösche den Flash-Speicherbereich

Eingabe:
DS = Segment des Eingabedateikopfteils
SI = Offset des Eingabedateikopfteils
ES = Segment des zusätzlichen Speichers, der durch das BIOS angefordert wurde
DI = Offset des zusätzlichen Speichers, der durch das BIOS angefordert wurde
CH = Status
0 = normaler Modus
1 = Wiederherstellungs-Modus
BX = Segment des Puffers für BIOS-Tabellen
DX = Offset des Puffers für BIOS-Tabellen

Ausgabe:
AH = Statusausgabe
0 = erfolgreiches Lesen

1 = ungültige Eingabe
2 = Löschfehler

AL = 01H Programmiere den Flash-Speicherbereich

Eingabe:
CH = Status
0 = normaler Modus
1 = Wiederherstellungs-Modus
DS = Segment des Eingabedateikopfteils
SI = Offset des Eingabedateikopfteils
ES = Segment des zusätzlichen Speichers, der durch das BIOS angefordert wurde
DI = Offset des zusätzlichen Speichers, der durch das BIOS angefordert wurde.
BX = Segment des Puffers für BIOS-Tabellen
DX = Offset des Puffers für BIOS-Tabellen

Ausgabe:
AH = Statusausgabe
0 = erfolgreiches Lesen
1 = ungültige Eingabe
3 = Bereich nicht gelöscht
4 = Überprüfungsfehler

Patentansprüche

1. Computersystem mit einem Prozessor (**101**) zur Ausführung von Verarbeitungslogik, **dadurch gekennzeichnet,**
daß in einem mit dem Prozessor gekoppelten nicht-flüchtigen Speicher (**103**) die Verarbeitungslogik des Betriebssystems und Daten gespeichert sind;
daß mit dem Prozessor (**101**) Mittel zum Lesen des Inhalts des nicht-flüchtigen Speichers, Mittel zum Löschen des Inhalts des nicht-flüchtigen Speichers und Mittel zum Programmieren des nicht-flüchtigen Speichers gekoppelt sind; und
daß das Computersystem Mittel zum Aktualisieren der Verarbeitungslogik des Betriebssystems während der Ausführung der Verarbeitungslogik des Betriebssystems aufweist.

2. Computersystem nach Anspruch 1, dadurch gekennzeichnet, daß der nicht-flüchtige Speicher (**103**) ein Flash-Speicherbauelement ist.

3. Computersystem nach Anspruch 1 oder 2, dadurch gekennzeichnet, daß der nicht-flüchtige Speicher (**103**) in mindestens zwei getrennt löschbare und programmierbare Abschnitte (**201**, **202**) eingeteilt ist, daß in einem ersten Abschnitt (**202**) Wiederherstellungs-Verarbeitungslogik und Daten gespeichert und daß in einem zweiten Abschnitt (**201**) Verarbeitungslogik und Daten des normalen Betriebssystems gespeichert sind.

4. Computersystem nach einem der Ansprüche 1 bis 3, dadurch gekennzeichnet, daß ein RAM (**102**) mit dem Prozessor (**101**) gekoppelt ist.

5. Computersystem nach einem der Ansprüche 1 bis 4, dadurch gekennzeichnet, daß die Verarbeitungslogik des Betriebssystems ein Basis-Eingabe-Ausgabe-System (BIOS) ist.

6. Computersystem nach einem der Ansprüche 1 bis 5, dadurch gekennzeichnet, daß die Mittel zum Aktualisieren ferner mit dem Prozessor (**101**) gekoppelte Mittel zum Auswählen einer Aktualisierungsart aufweisen und daß die Aktualisierungsart entweder eine normale oder eine Wiederherstellungs-Aktualisierungsart ist.

7. Computersystem nach Anspruch 6, dadurch gekennzeichnet, daß die normale Aktualisierung-Verarbeitungslogik arbeitsfähig in dem RAM (**102**) angeordnet ist und daß die Mittel zum Aktualisieren ferner Mittel zum Aktivieren der normalen Aktualisierungs-Verarbeitungslogik bei Auswahl der normalen Aktualisierungsart durch die Mittel zum Auswählen der Aktualisierungsart und Mittel zum Umprogrammieren des zweiten Abschnitts (**201**) bei Aktivierung der normalen Aktualisierungs-Verarbeitungslogik enthalten.

8. Computersystem nach Anspruch 7, dadurch gekennzeichnet, daß die Mittel zum Umprogrammieren ferner aufweisen:
eine geschützte Aktualisierungs-Verarbeitungslogik, die sich innerhalb der normalen, im zweiten Abschnitt gespeicherten Verarbeitungslogik des Betriebssystems befindet,
Mittel zum Übertragen der geschützten Aktualisierungs-Verarbeitungslogik zu dem RAM (**102**) und
Mittel zum Aktivieren der geschützten Aktualisierungs-Verarbeitungslogik, wobei die geschützte Aktualisierungs-Verarbeitungslogik die Programmierung des zweiten Abschnitts steuert.

9. Computersystem nach Anspruch 6, dadurch gekennzeichnet, daß die Mittel zum Aktualisieren ferner aufweisen:
Mittel zum Aktivieren der Wiederherstellungs-Verarbeitungslogik bei Auswahl der Wiederherstellungs-Aktualisierungsart durch die Mittel zum Auswählen der Aktualisierungsart und
Mittel zum Umprogrammieren des zweiten Abschnitts bei Aktivierung der Wiederherstellungs-Verarbeitungslogik.

10. Computersystem, nach einem der Ansprüche 3 bis 9, dadurch gekennzeichnet, daß der erste Abschnitt

(202) elektronisch gegen Löschen und/oder Verändern geschützt ist.

11. Computersystem mit einem Prozessor (101) zur Ausführung von Verarbeitungslogik, dadurch gekennzeichnet,
daß in einem mit dem Prozessor (101) gekoppelten nicht-flüchtigen Speicher (103) die Verarbeitungslogik des Betriebssystems und Daten gespeichert sind;
daß mit dem Prozessor (101) Mittel zum Lesen des Inhalts des nicht-flüchtigen Speichers, Mittel zum Löschen des Inhalts des nicht-flüchtigen Speichers und Mittel zum Programmieren des nicht-flüchtigen Speichers gekoppelt sind; und
daß das Computersystem Mittel zum Aktualisieren von in einem Bereich (203, 204, 205) des nicht-flüchtigen Speichers (103) angeordneter anwendungsspezifischer Verarbeitungslogik während der Ausführung der Verarbeitungslogik des Betriebssystems aufweist.

12. Computersystem nach Anspruch 11, dadurch gekennzeichnet, daß es ferner Mittel zum Freigeben einer Abtastung desjenigen Bereichs (203, 204, 205) des nicht-flüchtigen Speichers (103) aufweist, der anwendungsspezifische Verarbeitungslogik enthält.

13. Computersystem nach Anspruch 11 oder 12, dadurch gekennzeichnet, daß die anwendungsspezifische Verarbeitungslogik LAN-Verarbeitungslogik aufweist.

14. Computersystem nach einem der Ansprüche 11 bis 13, dadurch gekennzeichnet, daß die anwendungsspezifische Verarbeitungslogik SCSI-Verarbeitungslogik aufweist.

15. In Anwendung auf ein Computersystem mit einem Verarbeitungslogik ausführenden Prozessor und einem mit dem Prozessor gekoppelten nicht-flüchtigen Speicher, in welchem Verarbeitungslogik des Betriebssystems und Daten gespeichert werden, ein Verfahren zum Aktualisieren der Verarbeitungslogik eines Betriebssystems während der Ausführung der Verarbeitungslogik, dadurch gekennzeichnet,
daß entweder eine normale Aktualisierungsart oder eine Wiederherstellungs-Aktualisierungsart ausgewählt wird;
daß eine Wiederherstellungs-Verarbeitungslogik aktiviert wird, wenn die Wiederherstellungs-Aktualisierungsart ausgewählt wurde; und
ein erster Abschnitt des nicht-flüchtigen Speichers umprogrammiert wird, wenn die Wiederherstellungs-Verarbeitungslogik aktiviert wurde, wobei der Umprogrammierungsschritt ausgeführt wird, während die Verarbeitungslogik des Betriebssystems ausgeführt wird.

16. Verfahren nach Anspruch 15, dadurch gekennzeichnet, daß der Schritt des Umprogrammierens ferner folgende Schritte einschließt:
Übertragen geschützter Aktualisierungs-Verarbeitungslogik von der Wiederherstellungs-Verarbeitungslogik zum einem RAM; und
Aktivieren der geschützten Aktualisierungs-Verarbeitungslogik, wobei die geschützte Aktualisierungs-Verarbeitungslogik das Programmieren des ersten Abschnitts steuert.

17. Verfahren nach Anspruch 15, dadurch gekennzeichnet, daß folgende Schritte ausgeführt werden:
Aktivieren einer normalen Aktualisierungs-Verarbeitungslogik, wenn die normale Aktualisierungsart ausgewählt wurde; und
Umprogrammieren eines zweiten Abschnitts des nicht-flüchtigen Speichers, wenn die normale Aktualisierungs-Verarbeitungslogik aktiviert wurde, wobei der Schritt des Umprogrammierens eines zweiten Abschnitts ausgeführt wird, während die Verarbeitungslogik des Betriebssystems ausgeführt wird.

18. In Anwendung auf ein Computersystem mit einem Verarbeitungslogik ausführenden Prozessor und einem mit dem Prozessor gekoppelten nicht-flüchtigen Speicher, in dem Verarbeitungslogik des Betriebssystems und Daten gespeichert werden, ein Verfahren zum Aktualisieren von in einem Bereich des nicht-flüchtigen Speichers befindlicher anwendungsspezifischer Verarbeitungslogik während die Verarbeitungslogik des Betriebssystems ausgeführt wird, dadurch gekennzeichnet,
daß ein Bereich in dem nicht-flüchtigen Speicher zum Aktualisieren ausgewählt wird; und
daß der zum Aktualisieren ausgewählte Bereich des nicht-flüchtigen Speichers mit anwendungsspezifischer Verarbeitungslogik umprogrammiert wird, wobei das Umprogrammieren während der Ausführung der Verarbeitungslogik des Betriebssystems erfolgt.

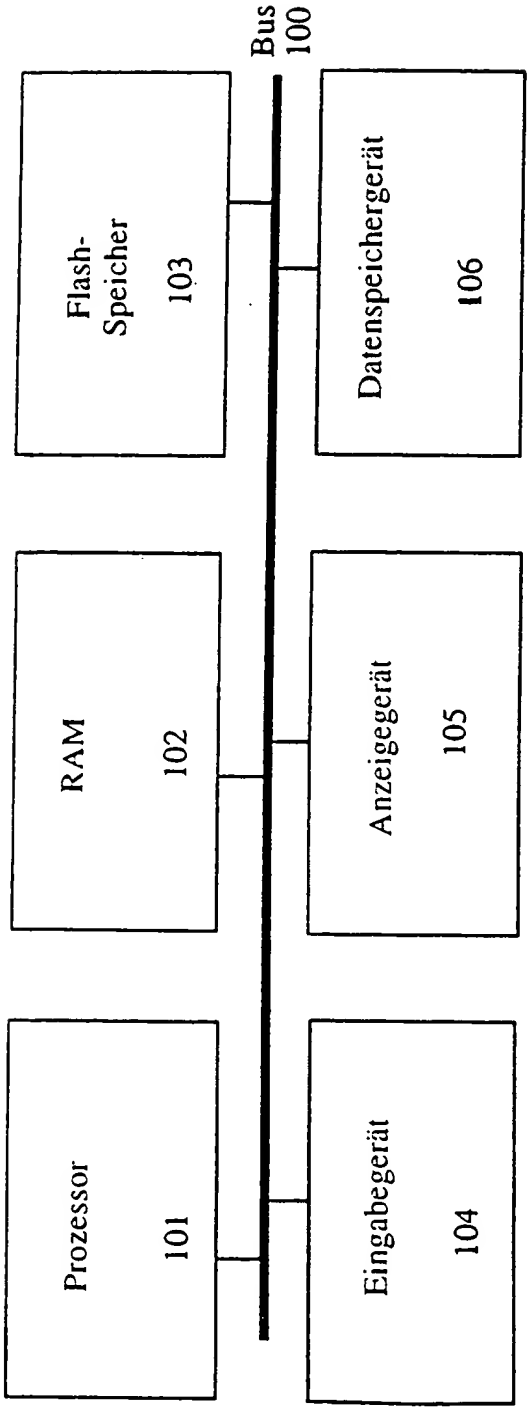
19. Verfahren nach Anspruch 18, dadurch gekennzeichnet, daß zusätzlich ein Schritt des Freigebens der Abtastung des Bereichs des anwendungsspezifische Verarbeitungslogik enthaltenden nicht-flüchtigen Speichers ausgeführt wird.

20. Verfahren nach Anspruch 18, dadurch gekennzeichnet, daß die anwendungsspezifische Verarbeitungslogik LAN-Verarbeitungslogik einschließt.

21. Verfahren nach Anspruch 18, dadurch gekennzeichnet, daß die anwendungsspezifische Verarbeitungslogik SCSI-Verarbeitungslogik einschließt.

Hierzu 6 Seite(n) Zeichnungen

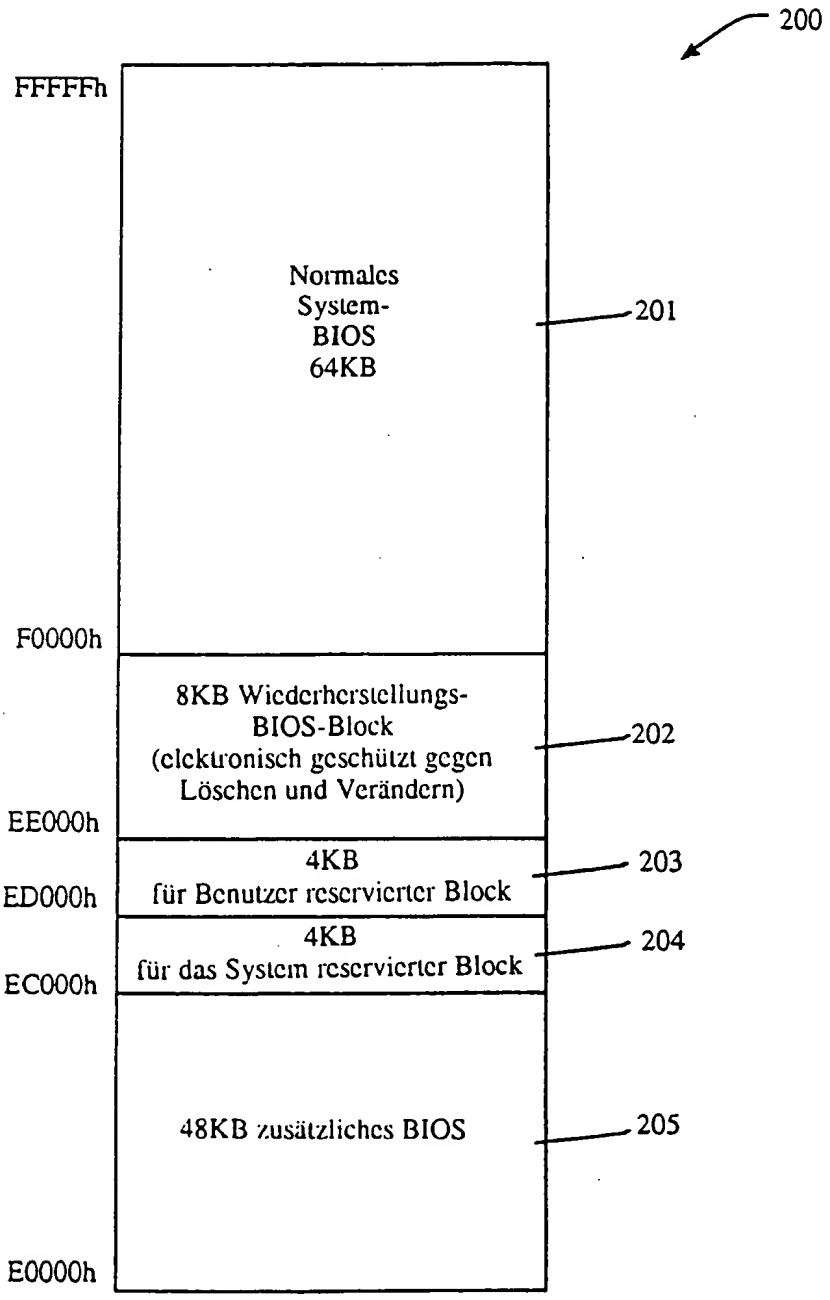
Nummer: DE 42 14 184 A1
Int. Cl.[5]: G 06 F 12/00
Offenlegungstag: 12. November 1992



Figur 1

Nummer: DE 42 14 184 A1
Int. Cl.[5]: G 06 F 12/00
Offenlegungstag: 12. November 1992

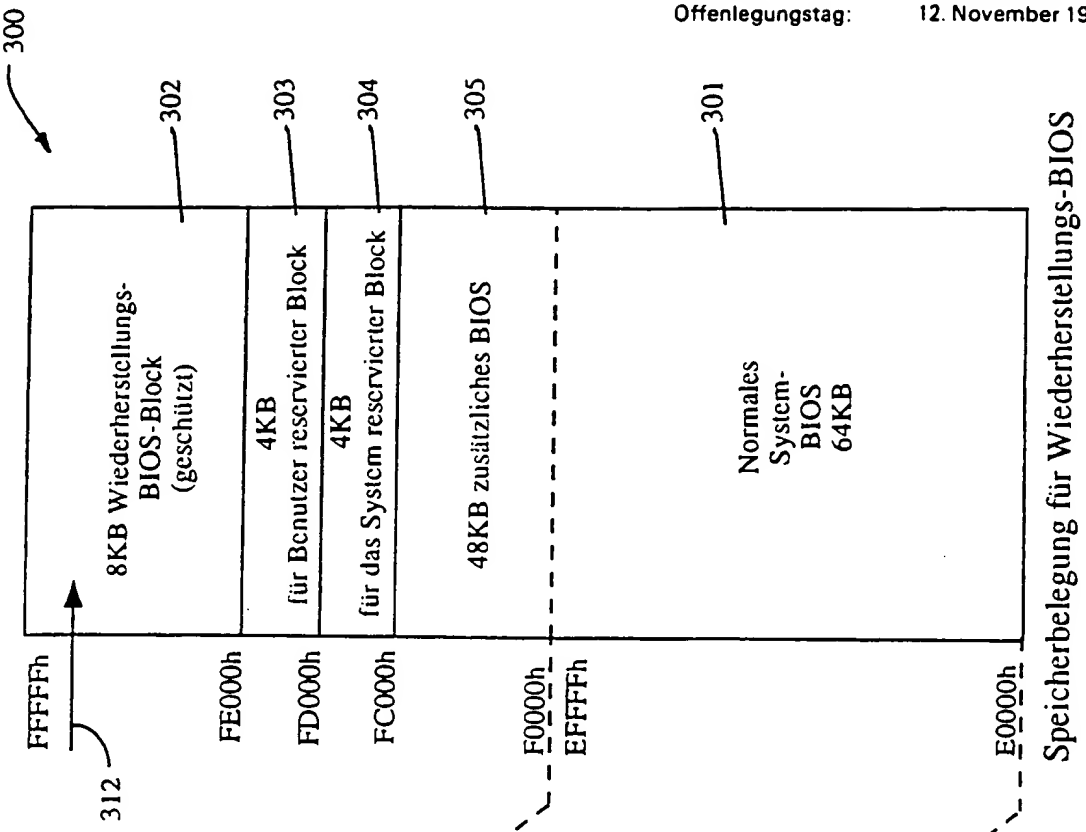
FIGUR 2



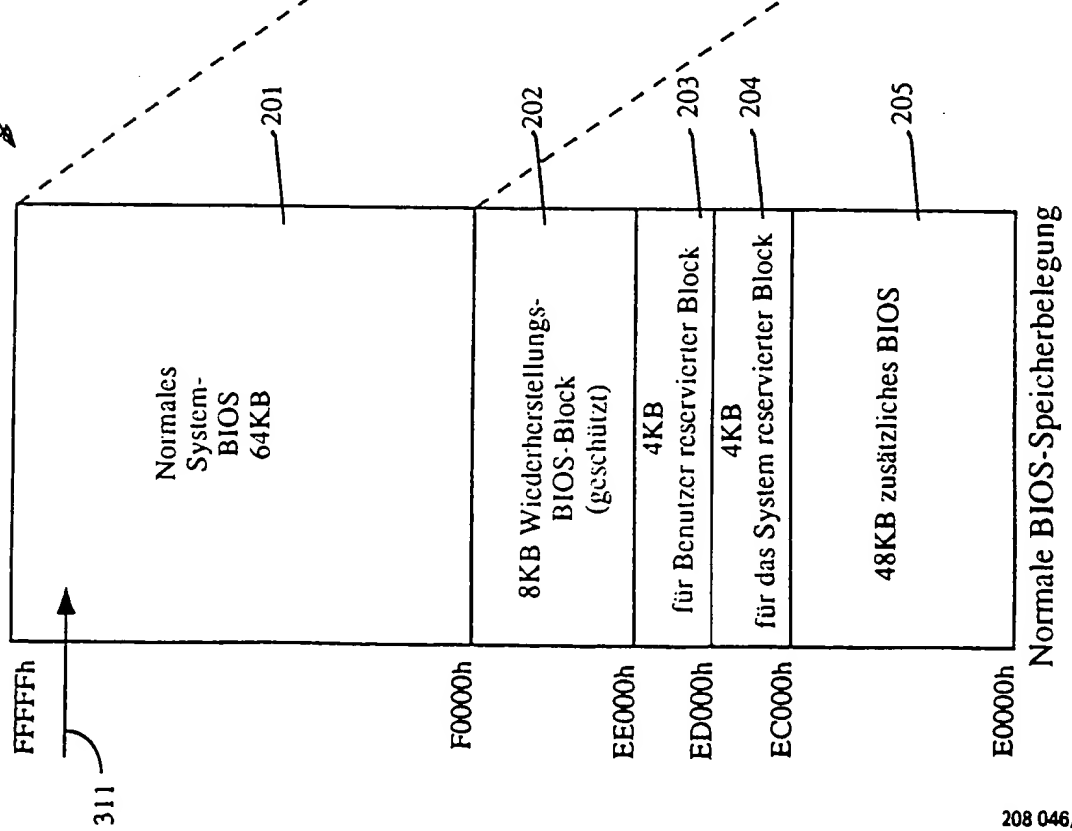
Normale BIOS-Speicherbelegung

Nummer: DE 42 14 184 A1
Int. Cl.[5]: G 06 F 12/00
Offenlegungstag: 12. November 1992

FIGUR 3B

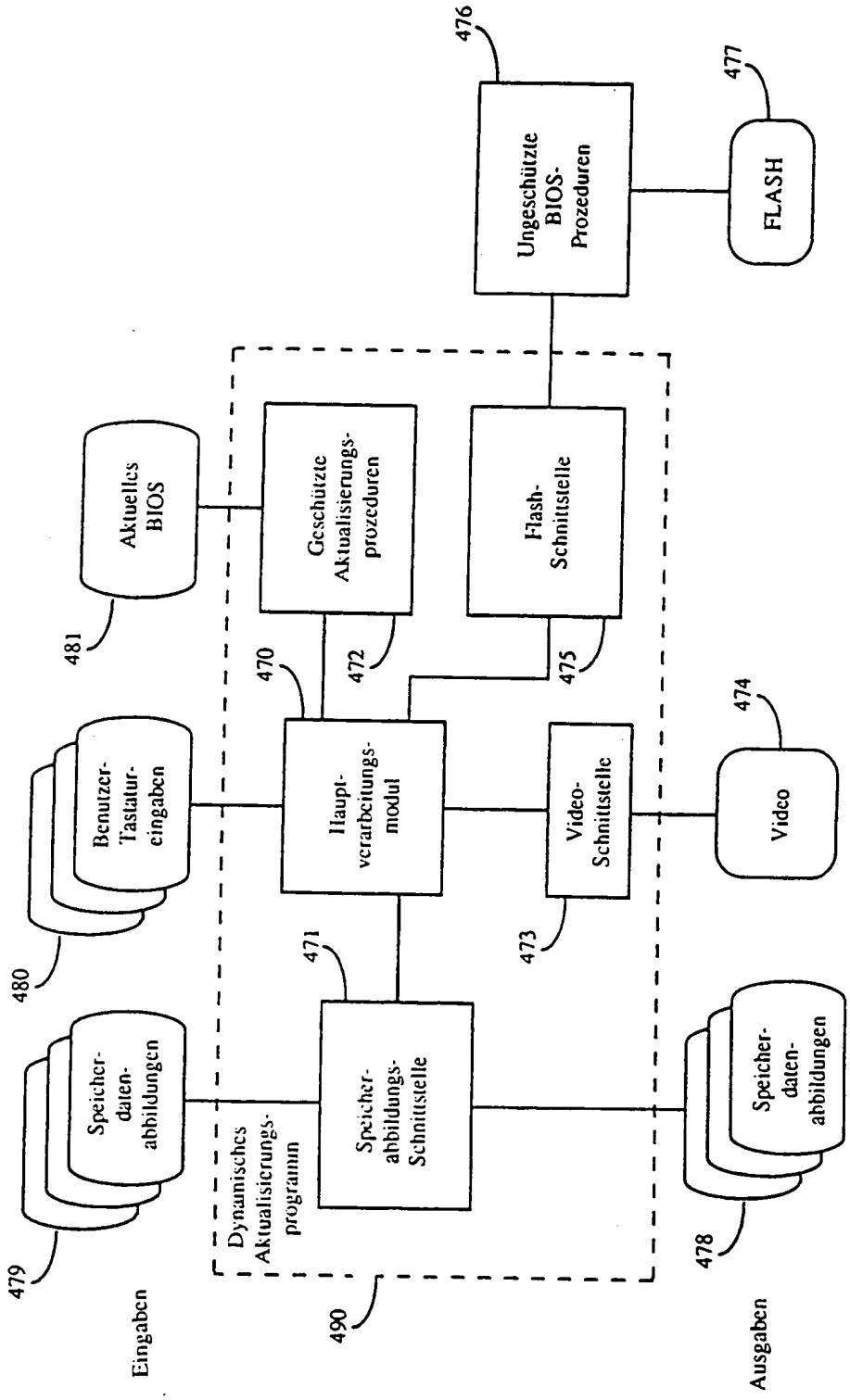


FIGUR 3A



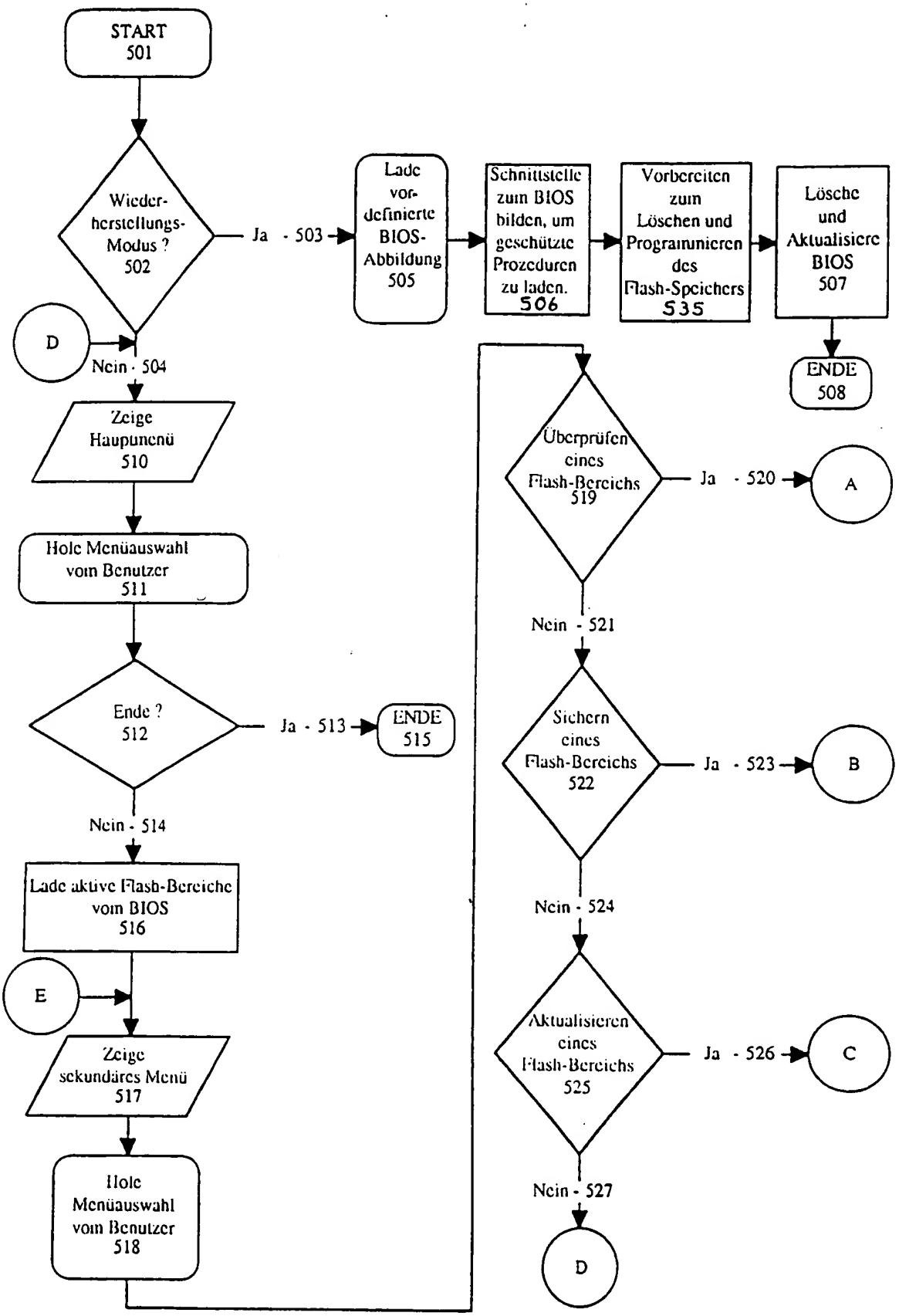
Nummer: DE 42 14 184 A1
Int. Cl.[5]: G 06 F 12/00
Offenlegungstag: 12. November 1992

FIGUR 4



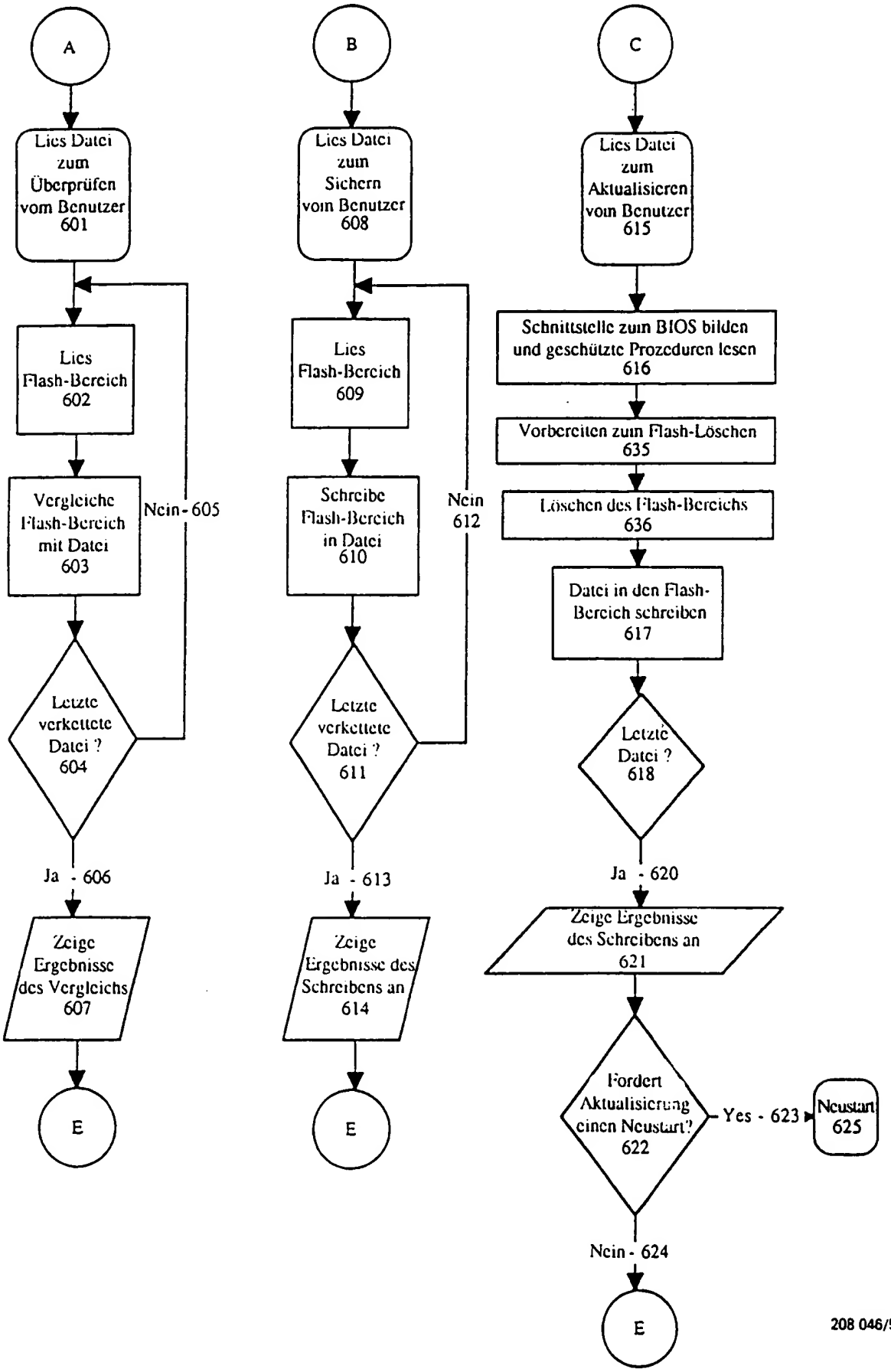
Nummer: DE 42 14 184 A1
Int. Cl.[5]: G 06 F 12/00
Offenlegungstag: 12. November 1992

FIGUR 5



Nummer: DE 42 14 184 A1
Int. Cl.[5]: G 06 F 12/00
Offenlegungstag: 12. November 1992

FIGUR 6



208 046/501

ENGLISH TRANSLATION OF REFERENCE NO. 220

US005579522A

United States Patent [19]

Christeson et al.

[11] Patent Number: 5,579,522

[45] Date of Patent: Nov. 26, 1996

[54] **DYNAMIC NON-VOLATILE MEMORY UPDATE IN A COMPUTER SYSTEM**

[75] Inventors: **Orville H. Christeson**, Portland; **Douglas L. Gabel**, Aloha; **Sean T. Murphy**, Portland, all of Oreg.

[73] Assignee: **Intel Corporation**, Santa Clara, Calif.

[21] Appl. No.: **505,995**

[22] Filed: **Jul. 24, 1995**

Related U.S. Application Data

[63] Continuation of Ser. No. 135,310, Oct. 12, 1993, which is a continuation of Ser. No. 695,952, May 6, 1991, abandoned.

[51] Int. Cl.[6] **G06F 9/06**

[52] U.S. Cl. **395/652**; 364/280.2; 364/280.9; 364/DIG. 1

[58] Field of Search 395/700; 364/280.2, 364/280.9

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,153,937	5/1979	Poland	364/706
4,290,104	9/1981	Holtey et al.	395/400
4,374,417	2/1983	Bradley et al.	395/400
4,441,155	4/1984	Fletcher et al.	395/400
4,443,847	4/1984	Bradley et al.	395/425
4,608,632	8/1986	Kummer	395/425
4,763,333	8/1988	Byrd	371/66
4,799,145	1/1989	Goss et al.	395/700
4,831,522	5/1989	Henderson et al.	395/425
4,862,349	8/1989	Foreman et al.	395/700
5,034,915	7/1991	Styma et al.	395/775
5,053,990	10/1991	Kreifels	364/900
5,117,492	5/1992	Nash	395/400

(List continued on next page.)

OTHER PUBLICATIONS

Waite et al., "CP/M Bible", 1983, pp. 5–22 and 100.

Glass, Brett, "The IBM PC BIOS", Byte, Apr. 1989, pp. 303–310.

Venditto, Gus, "Pipeline", PC Magazine V.9, N.3, Feb. 1990, pp. 1–3.

Machrone, Bill, "Bill Machrone", PC Magazine V.9, N.7, Apr. 1990, pp. 1–2 .

Venditto, Gus, "Intel's flash memory poised to give laptops their next great leap", PC Magazine V. 9, N. 14, Aug. 1990, pp. 1–3.

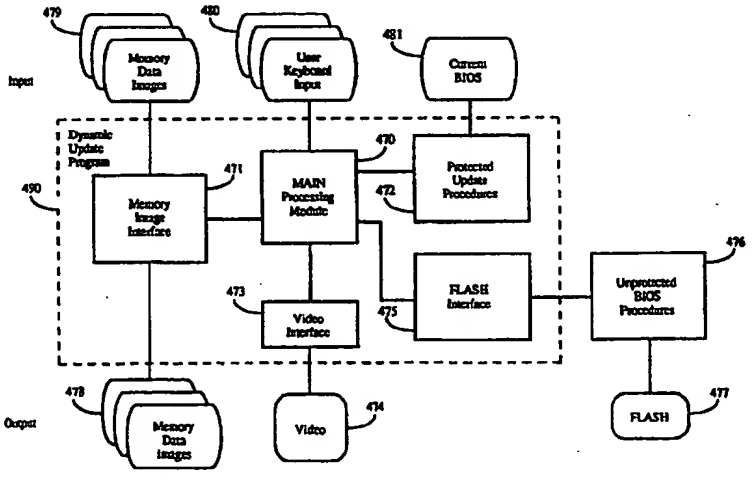
(List continued on next page.)

Primary Examiner—Kevin A. Kriess
Assistant Examiner—Dennis M. Butler
Attorney, Agent, or Firm—Blakely, Sokoloff, Taylor & Zafman

[57] **ABSTRACT**

A computer system wherein a portion of code/data stored in a non-volatile memory device can be dynamically modified or updated without removing any covers or parts from the computer system. The computer system of the preferred embodiment includes a flash memory component coupled to a computer system bus for storing non-volatile code and data. Using the present invention, the contents of a portion of the flash memory may be replaced, modified, updated, or reprogrammed without the need for removing and/or replacing any computer system hardware components. The flash memory device used in the preferred embodiment contains four separately erasable/programmable non-symmetrical blocks of memory. One of these four blocks may be electronically locked to prevent erasure or modification of its contents once it is installed. This configuration allows the processing logic of the computer system to update or modify any selected block of memory without affecting the contents of other blocks. One memory block contains a normal BIOS. An electronically protected flash memory area is used for storage of a recovery BIOS which is used for recovery operations. The present invention also includes hardware for selecting one of the two available update modes: normal or recovery. Thus, using a mode selection apparatus, either a normal system BIOS or a recovery BIOS may be activated.

25 Claims, 6 Drawing Sheets



U.S. PATENT DOCUMENTS

5,126,808	6/1992	Montalvo et al.	357/23.5
5,134,580	7/1992	Bertram et al.	395/650
5,136,713	8/1992	Bealkowski et al.	395/700
5,142,680	8/1992	Ottman et al.	395/700
5,210,875	5/1993	Bealkowski et al.	395/700
5,257,380	10/1993	Lang	395/700
5,371,876	12/1994	Ewertz et al.	395/425
5,388,267	2/1995	Chan et al.	395/700

OTHER PUBLICATIONS

Wharton, John H., "FLASH! memory technology marches on", Microprocessor Report, Aug. 1990, pp. 1–4.

Levy, Markus A., "Designing with Flash Memory", Circuit Cellar Ink Dec. 1990, pp. 50–58.

Jex, Jerry, "Flash Memory BIOS For PC and Notebook Computers", IEEE, 1990, pp. 692–695.

Waite et al., "Soul of CP/M", Howard W. Sams and Co., 1983, pp. 2, 7–10, 177–182 and 279–322.

Weber, S., *Look Out EPROM's, Here Comes Flash*, T.O.C. and 44, 46, 50; ELECTRONICS Magazine (Nov., 1990).

Shandle, J., *Laptop Vendors Join the Flash Bandwagon*, T.O.C. and 52–53; ELECTRONICS Magazine (Nov., 1990).

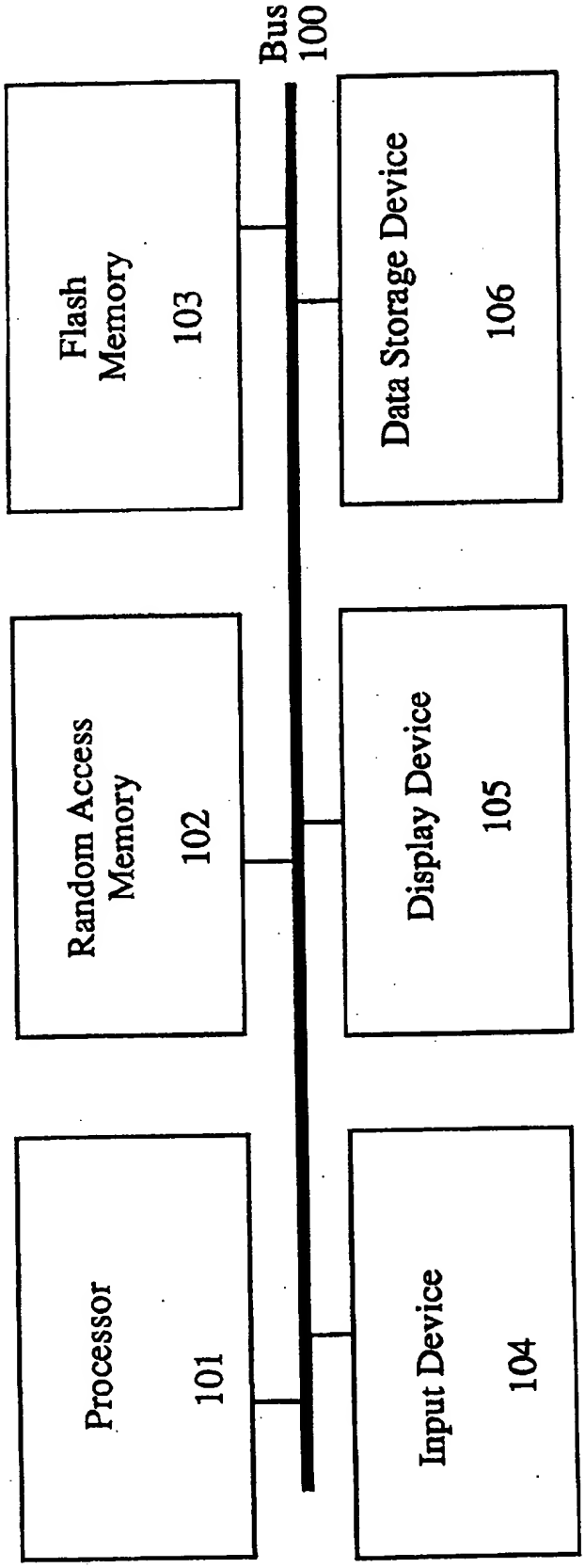
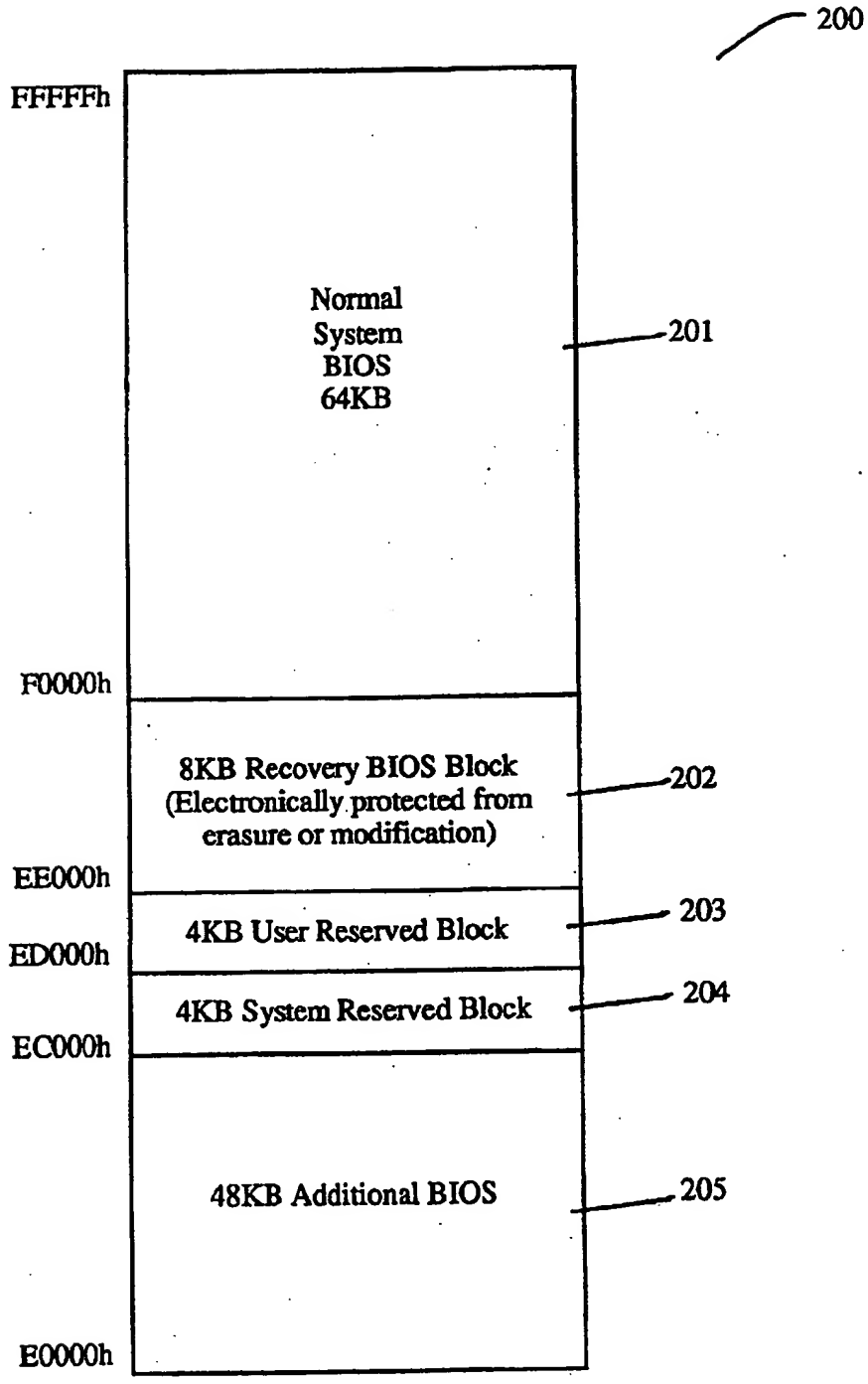

Bus
100
Flash Memory
103
Data Storage Device
106
Random Access Memory
102
Display Device
105
Processor
101
Input Device
104

Figure 1

FIGURE 2



Normal BIOS Map

FIGURE 3B

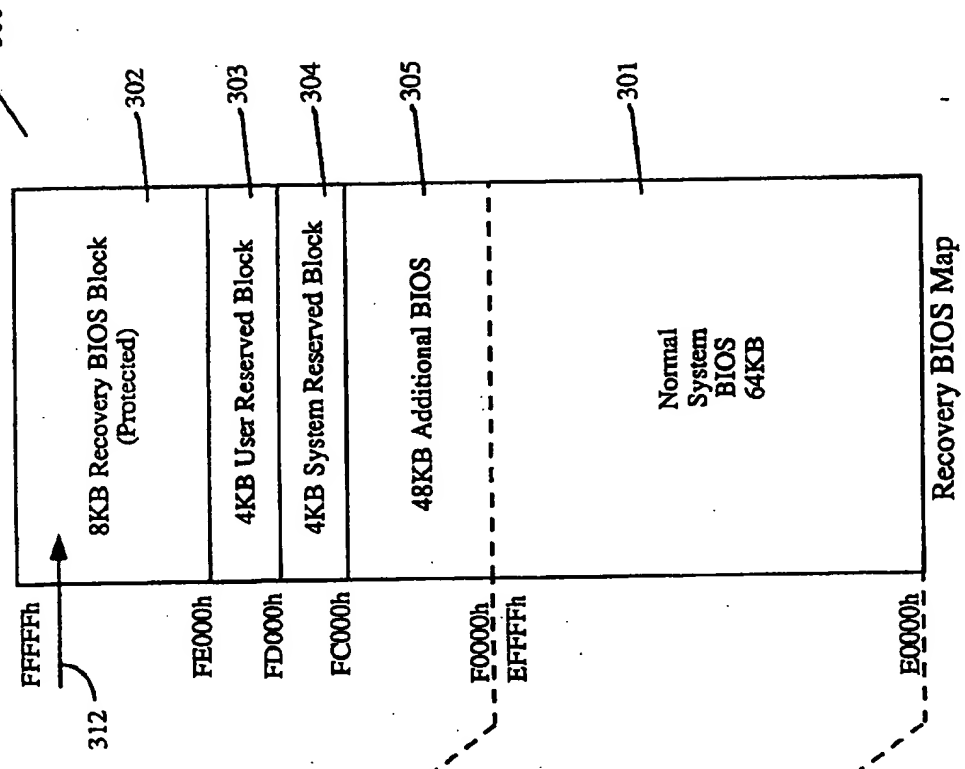


FIGURE 3A

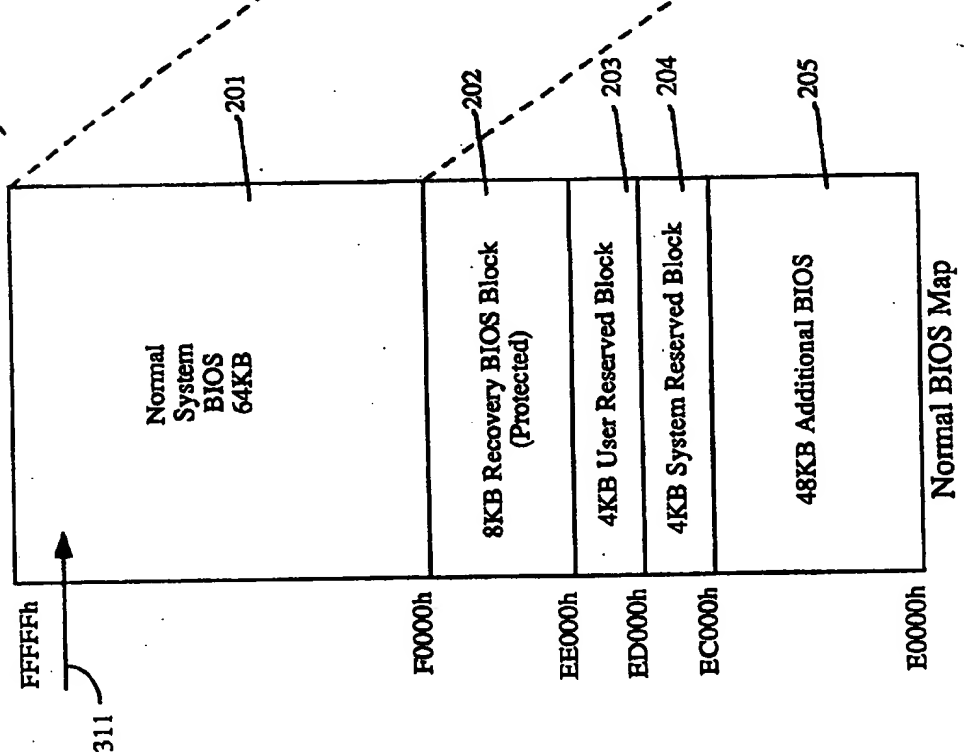


FIGURE 4

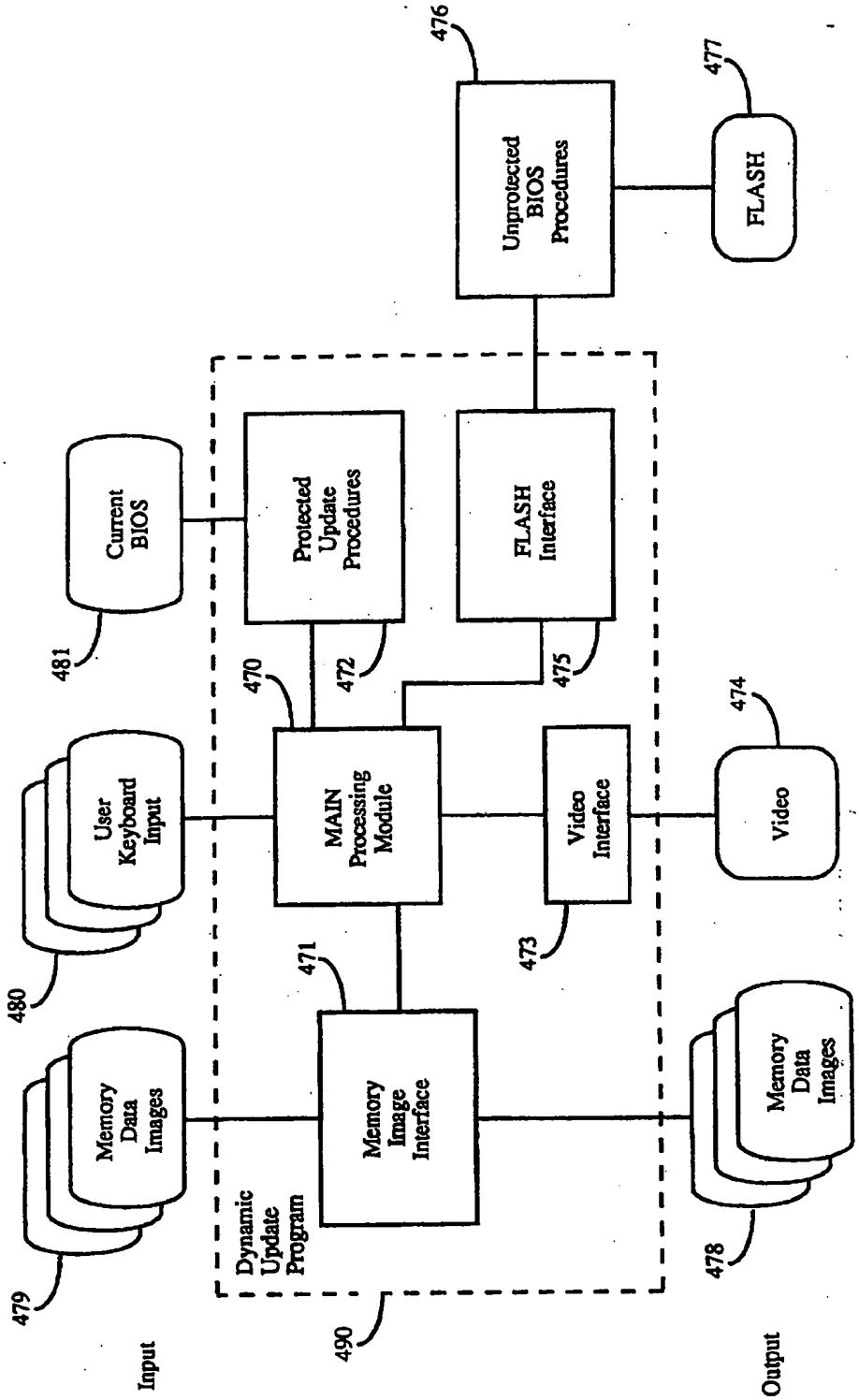

476
Unprotected BIOS Procedures
477
FLASH
Current BIOS
Protected Update Procedures
FLASH Interface
481
470
472
475
User Keyboard Input
MAIN Processing Module
Video Interface
474
Video
480
471
473
Memory Data Images
Memory Image Interface
Memory Data Images
Dynamic Update Program
479
490
478
Input
Output

FIGURE 5

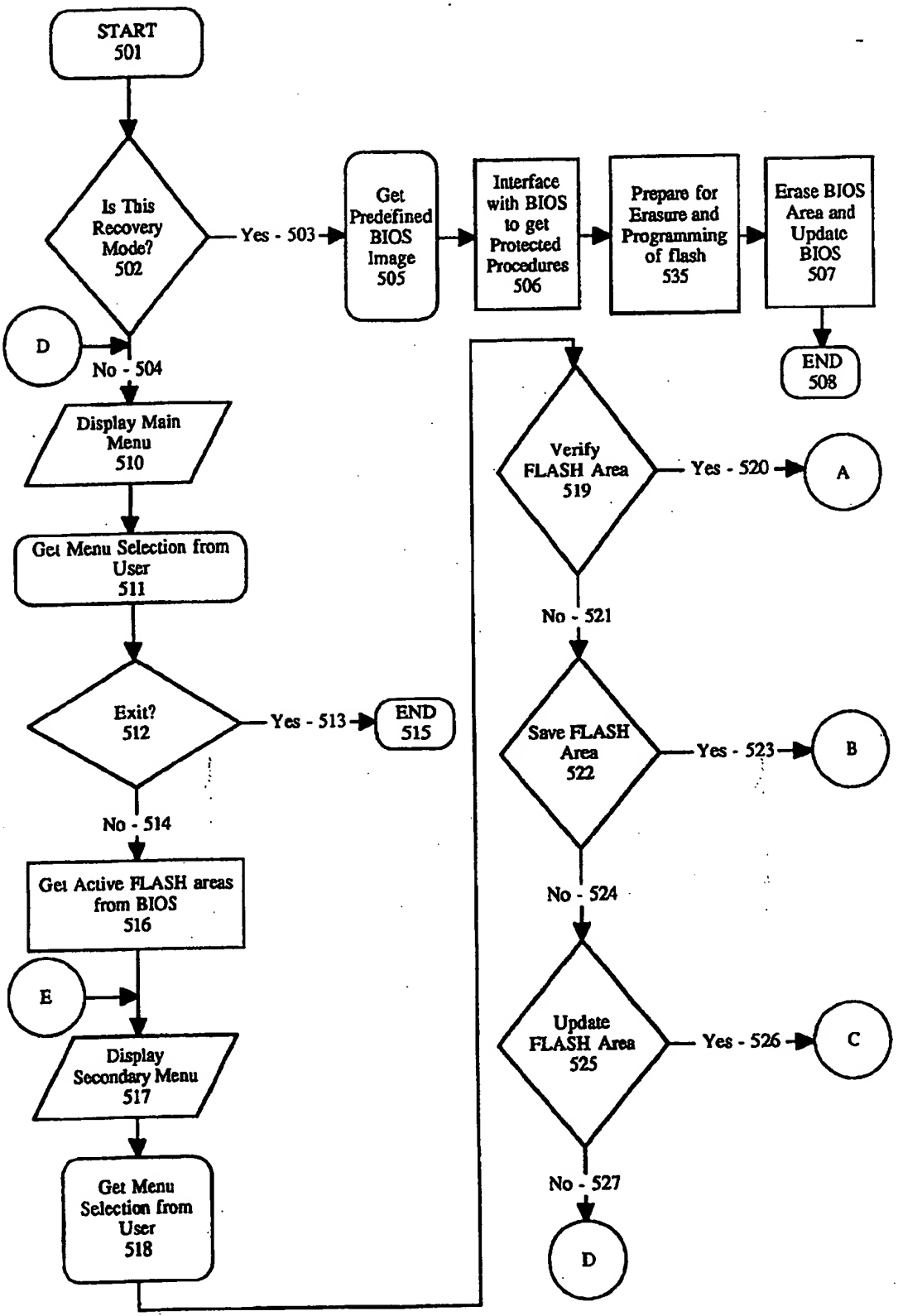
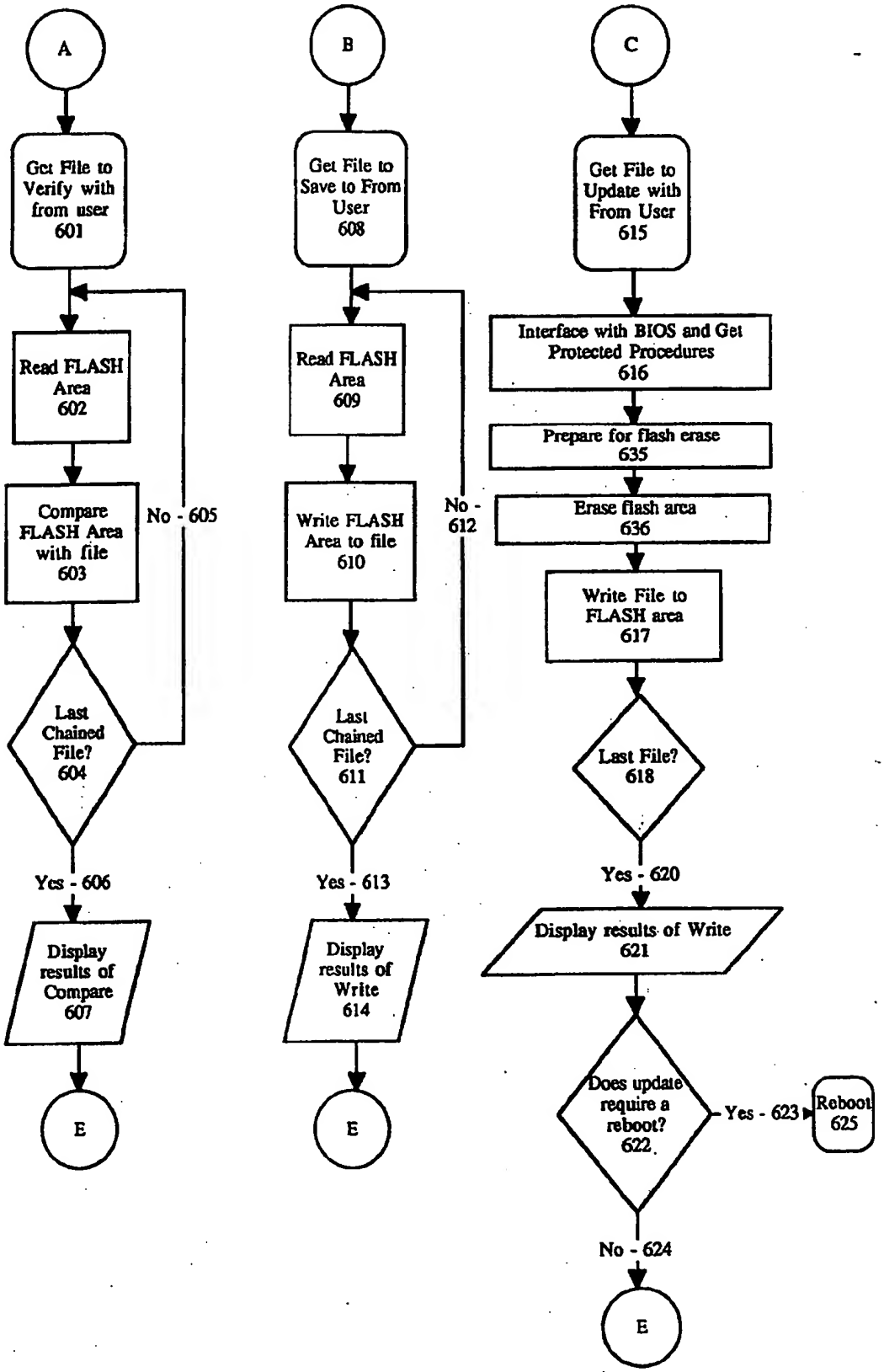

START
501
Is This Recovery Mode?
502
Yes - 503
Get Predefined BIOS Image
505
Interface with BIOS to get Protected Procedures
506
Prepare for Erasure and Programming of flash
535
Erase BIOS Area and Update BIOS
507
END
508
D
No - 504
Display Main Menu
510
Get Menu Selection from User
511
Exit?
512
Yes - 513
END
515
No - 514
Get Active FLASH areas from BIOS
516
E
Display Secondary Menu
517
Get Menu Selection from User
518
Verify FLASH Area
519
Yes - 520
A
No - 521
Save FLASH Area
522
Yes - 523
B
No - 524
Update FLASH Area
525
Yes - 526
C
No - 527
D

FIGURE 6


A
Get File to Verify with from user 601
Read FLASH Area 602
Compare FLASH Area with file 603
No - 605
Last Chained File? 604
Yes - 606
Display results of Compare 607
E
B
Get File to Save to From User 608
Read FLASH Area 609
Write FLASH Area to file 610
No - 612
Last Chained File? 611
Yes - 613
Display results of Write 614
E
C
Get File to Update with From User 615
Interface with BIOS and Get Protected Procedures 616
Prepare for flash erase 635
Erase flash area 636
Write File to FLASH area 617
Last File? 618
Yes - 620
Display results of Write 621
Does update require a reboot? 622
Yes - 623
Reboot 625
No - 624
E

DYNAMIC NON-VOLATILE MEMORY UPDATE IN A COMPUTER SYSTEM

This is a continuation of application Ser. No. 08/135,310, filed Oct. 12, 1993, which is a continuation of application Ser. No. 07/695,952, filed May 6, 1991, now abandoned.

FIELD OF THE INVENTION

The present invention relates to the field of computer systems. Specifically, the present invention relates to the field of computer system architectures incorporating a non-volatile form of basic operating system processing logic.

BACKGROUND OF THE INVENTION

Many prior art computer systems are typically configured at a minimum with a processor, a random access memory device, and a read only memory device. Some systems, such as a variety of calculators, may operate with only a processor and a read only memory device. Read only memory devices (ROM) provide a non-volatile form of memory that is not destroyed when power is removed from the computer system.

Prior art computer systems are typically bootstrapped (i.e. power up initialized) using the processing logic (i.e. firmware) stored within the read only memory device internal to the computer system. Since the read only memory device is non-volatile, the firmware within ROM is guaranteed to contain valid data or instructions; thus, the prior art computer system can be reliably bootstrapped using firmware within ROM. Many computer systems have successfully used this technique. One such system is the IBM Personal Computer (PC) developed by the IBM Corporation of Armonk, New York. Prior art versions of the IBM PC use read only memory devices for storage of firmware or a basic input/output system (BIOS) software program. The BIOS is operating system processing logic that provides the lowest level of software control over the hardware and resources of the computer system. ROM storage may also be used for non-volatile retention of network configuration data or application specific data. ROM devices in the prior art include basic read only memory devices (ROM), programmable read only memory devices (PROM), and eraseable programmable read only memory devices (EPROM).

Although ROM based computer systems have been very successful in the prior art, a number of problems exist with the use of these devices in a computer system. Read only memory devices must be programmed with a BIOS and/or data prior to being placed into the system during production and assembly of the computer system. Often the BIOS ROM is installed on a system circuit board within the computer housing. In order to replace, modify, or update firmware in a ROM based computer system, the computer housing must be removed and ROM devices on a system circuit board internal to the computer system must be disconnected and replaced or reprogrammed. This invasive ROM replacement and reprogramming procedure is disadvantageous for a number of reasons. First, the ROM replacement operation typically must be performed manually by qualified field service or computer repair personnel; thus, the operation tends to be expensive and time consuming. Secondly, even qualified technical service personnel may introduce problems during the ROM replace operation. If solder connections are necessary, existing connections may be damaged or weakened in the process. Also, electrostatic discharge may inadvertently cause damage to other components on the circuit board during the ROM replace operation. Thirdly, ROM based computer systems are not easily customized for specific applications. Such customization includes modifications for operation in non-English speaking countries. In order to customize a computer system by storing language-specific data in non-volatile memory, a user must program and install a read only memory device on a circuit board in the computer system. Because of the inconvenient ROM installation procedure in the prior art, a user is unable or less likely to customize his/her computer system.

Thus a better means for storing and updating non-volatile code and/or data in a computer system is needed.

SUMMARY OF THE INVENTION

The present invention is a computer system wherein a portion of code/data stored in a non-volatile memory device can be dynamically modified or updated without removing any covers or parts from the computer system. The computer system of the preferred embodiment comprises a bus for communicating information, a processor coupled with the bus for processing information, a random access memory device coupled with the bus for storing information and instructions for the processor, an input device such as an alpha numeric input device or a cursor control device coupled to the bus for communicating information and command selections to the processor, a display device coupled to the bus for displaying information to a computer user, and a data storage device such as a magnetic disk and disk drive coupled with the bus for storing information and instructions. In addition, the computer system of the preferred embodiment includes a flash memory component coupled to the bus for storing non-volatile code and data. Devices other than flash memory may be used for storing non-volatile code and data. Using the present invention, the contents of the flash memory may be replaced, modified, updated, or reprogrammed without the need for removing and/or replacing any computer system hardware components.

The flash memory device used in the preferred embodiment contains four separately erasable/programmable non-symmetrical blocks of memory. One of these four blocks may be electronically locked to prevent erasure or modification of its contents once it is installed. This configuration allows the processing logic of the computer system to update or modify any selected block of memory without affecting the contents of other blocks. One memory block contains a normal BIOS. The BIOS comprises processing logic instructions that are executed by the processor. An additional BIOS region can be used to extend the system BIOS memory area. An electronically protected (i.e. locked) flash memory area is used for storage of a recovery BIOS which is used for recovery operations. Each of these separately programmable regions of the flash memory may be modified or updated using the dynamic update mechanism of the present invention.

In order to prevent an aborted BIOS update from rendering the computer system non-functional, the update procedure of the present invention operates in two distinct user environments: a normal update mode and a recovery update mode. In normal update mode, the keyboard and video services of the computer system are available for receiving command selections and displaying results. The normal update mode also provides functionality to save, verify, or update areas of flash memory in addition to the BIOS areas. In recovery update mode, only the system BIOS areas can be

updated. Recovery update mode is used when a user cannot boot the system because the normal system BIOS has been corrupted either following a power failure during a normal BIOS update or for some other reason.

A dynamic BIOS update program contains most of the processing logic of the present invention. The remaining portions of the processing logic of the present invention reside in the BIOS image itself. The dynamic update processing logic includes logic for handling a normal update mode and the recovery update mode.

The present invention also includes hardware means for selecting one of these two available update modes. In the preferred embodiment, this means of selecting an update mode is implemented as a jumper on a circuit board in the computer system.

Once the recovery mode is set using the jumper, the processor of the computer system may then be power-up initialized or reset. Upon power up or reset, the processor jumps to a location within the protected recovery BIOS block. In this manner, the flash device memory map may be reconfigured in a recovery mode by activating recovery mode processing logic residing in recovery BIOS block. Thus, using the mode selection means, either a normal system BIOS or a recovery BIOS may be activated.

When executing out of normal BIOS, the dynamic update program displays a menu of options for user selection. These options include: verification of a flash memory area, saving a selected flash memory area, updating a flash memory area, and exit. Using these command options, the flash memory areas that may be updated might include the normal system BIOS area, a user reserved area, a local area network (LAN) BIOS area, a SCSI BIOS area, a video data area, other hardware or software specific BIOS or data areas, or any other application specific processing logic in an area of flash memory.

It is therefore an object of the present invention to provide a means for storing non-volatile code and data in a computer system where non-volatile code and data may be updated without removing any parts from the computer system. It is a further object of the present invention to provide a means for updating a selected portion of non-volatile memory while leaving other portions unmodified. It is a further object of the present invention to provide a means for recovery in the event of a failure during the normal non-volatile memory update procedure. It is a further object of the present invention to provide a means for saving the current contents of non-volatile memory. It is a further object of the present invention to provide a means for verifying the content of code and/or data currently residing in non-volatile memory. It is a further object of the present invention to provide a means for embedding update control software within system BIOS such that the update control software may later be extracted and used for controlling the non-volatile memory update procedure.

These and other objects of the present invention will become apparent as presented and described in the following detailed description of the preferred embodiment.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is an illustration of a computer system of the present invention.

FIG. **2** is an illustration of a memory map of the BIOS in the flash memory device used in the preferred embodiment.

FIG. **3***a* is a memory map of the BIOS during normal operation.

FIG. **3***b* is an illustration of the memory map of the BIOS during recovery mode.

FIG. **4** is a block diagram of the processing logic architecture of dynamic BIOS update mechanism.

FIGS. **5** and **6** are flow charts of the dynamic BIOS update processing logic of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention is a computer system wherein a portion of the non-volatile memory may be modified or updated without removal of system hardware components. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be apparent to one of ordinary skill in the art that these specific details need not be used to practice the present invention. In other instances, well known structures, circuits and interfaces have not been shown in detail in order not to unnecessarily obscure the present invention.

Referring to FIG. **1**, a block diagram of the computer system of the present invention is illustrated. The preferred embodiment of the present invention is implemented using an 80386 microprocessor manufactured by the Assignee of the present invention. It will be apparent to those with ordinary skill in the art, however, that alternative processors and computer system architectures may be employed. In general, such computer systems as illustrated by FIG. **1** comprise a bus **100** for communicating information, a processor **101** coupled with the bus for processing information, a random access memory device **102** coupled with the bus **100** for storing information and instructions for the processor **101**, an input device **104** such as an alpha numeric input device or a cursor control device coupled to the bus **100** for communicating information and command selections to the processor **101**, a display device **105** coupled to the bus **100** for displaying information to a computer user, and a data storage device such as a magnetic disk and disk drive coupled with the bus **100** for storing information and instructions.

In addition, the computer system of the preferred embodiment includes flash memory component **103** coupled to the bus **100** for storing non-volatile code and data. Flash memory component **103** provides a non-volatile form of memory that is not destroyed when power is removed from the computer system; however, the contents of flash memory may be erased and reprogrammed. Flash memory devices are well known in the art. The basic input/output processing logic (BIOS) of the computer system is stored in flash memory **103**. In addition, other system software and application specific parameters may also be stored in flash memory **103**. For example, portions of flash memory may be used for storage of local area network (LAN) processing logic or Small Computer Standard Interface (SCSI) processing logic. The following sections describe how portions of the flash memory **103** may be replaced, modified, or reprogrammed (i.e. updated) without the need for removing and/or replacing any computer system hardware components.

Several types of non-volatile memory devices currently existing in the art may be reprogrammed without removing the device from a circuit board on which the device is installed. One class of reprogrammable non-volatile memory devices is flash memory. Several different types of flash memory devices exist in the art. Using a dedicated set

of electrical signals, the contents of flash memory may be erased and reprogrammed with new data. Many prior art flash memory devices only allow complete erasure and reprogramming of all memory locations of the device. Other flash memory devices, however, are partitioned into separately erasable and programmable blocks of memory in a single flash memory device. In the preferred embodiment of the present invention, such a partitioned flash memory device is used. In the preferred embodiment, a flash memory device denoted 28F001BT is used. The 28F001BT flash memory device is a 1M bit memory device manufactured by the Assignee of the present invention. It will be apparent to those skilled in the art that other forms of reprogrammable non-volatile memory devices may be used with the invention taught herein. One example of such a non-flash device is an electrically erasable programmable read only memory (EEPROM).

The flash memory device used in the preferred embodiment contains four separately erasable/programmable non-symmetrical blocks of memory. One of these four blocks may be electronically locked to prevent erasure or modification of its contents once it is installed. This configuration allows the processing logic of the computer system to update or modify any selected block of memory without affecting the contents of other blocks. Referring to FIG. 2, several separately erasable/programmable non-symmetric blocks of the flash memory device of the preferred embodiment are illustrated. FIG. 2 depicts a normal BIOS memory map 200 of the contents of flash memory. The BIOS comprises processing logic instructions that are executed by the processor. In the preferred embodiment of the present invention, the processor of the computer system is an 80386 microprocessor. When the computer system of the preferred embodiment is first powered up, the processor jumps to an initial location FFFF0H and starts execution. In order to handle the initialization of the computer system, the active system BIOS must therefore include the location FFFF0H. As indicated by the normal BIOS map 200 illustrated in FIG. 2, the processor jumps to a location within a normal system BIOS 201 on power up or reset of the processor. Processing logic within region 201 may be used to handle normal initialization and control functions of the computer system. Additional BIOS region 205 can be used to extend the system BIOS memory area.

Electronically protected (i.e. locked) flash memory area 202 is an area for storage of a recovery BIOS used for recovery operations. The recovery operation is described in the sections below. Separately programmable area 203 is a memory area reserved for use by a particular user or application. This area may be used to customize the operation of the computer system or to enhance the functionality of the system BIOS. As described below, separately programmable area 203 may contain LAN processing logic, SCSI processing logic, video data or control logic, other hardware or software specific logic or data, or any other application specific processing logic in an area of flash memory. If separately programmable area 203 contains LAN or SCSI logic, the system BIOS is enabled to scan these additional flash areas in order to boot load the computer system from a network or other external device. Separately programmable area 204 is an area reserved for system use. This area may be an additional overflow area for normal system BIOS. Each of these separately programmable regions of the flash memory may be modified or updated using the dynamic update mechanism of the present invention as described below.

One disadvantage of using a flash memory device for storage of a BIOS is the possibility of a power failure or other error during the update process. The resulting incomplete or corrupted BIOS would render the system non-functional. In order to restore operation of the computer system after an aborted BIOS update, parts of the computer system would have to be removed and reprogrammed. Removal of flash memory components may be further complicated if surface mount flash devices are used.

In order to prevent an aborted BIOS update from rendering the computer system non-functional, the update procedure of the present invention operates in two distinct user environments: a normal update mode and a recovery update mode. The major difference between these two environments is the user interface. In normal update mode, the keyboard and video services of the computer system are available for receiving command selections and displaying results. Using the robust normal update user interface, the user may specify a save, verify, or update flash memory operation. The user may also specify the data file to retrieve or save from various sources such as a floppy disk drive, hard disk drive, network, or modem. The data contained within a retrieved or saved file may be a BIOS image, data image, or other memory image representation of the contents of a non-volatile or flash memory area. The normal update mode also provides functionality to save, verify, or update other areas of flash memory in addition to the BIOS areas. In recovery update mode, only the system BIOS areas can be updated. Recovery update mode is used when a user cannot boot the system because the normal system BIOS has been corrupted either following a power failure during a normal BIOS update or for some other reason. In this situation, a separate set of processing logic residing in the protected recovery BIOS block 202 illustrated in FIG. 2 is executed strictly for the purpose of updating the corrupted normal system BIOS area 201. In order to reduce the memory area consumed by the recovery BIOS processing logic, only a minimal level of functionality is provided and no user interface capability is supported. The keyboard and video services are not available and predetermined actions must be performed by the recovery BIOS without user interaction. Audible beep codes issued from a speaker of the computer system indicate the status of the programming of the BIOS area.

In the normal operation of the computer system of the preferred embodiment, processor 101 initially executes instructions out of the normal system BIOS region 201 of flash memory 103. Subsequently, a full-featured operating system may be read from data storage device 106 into random access memory (RAM) 102 and executed from RAM 102. Code in system BIOS 201 may also be used for initialization and normal access to input device 104, display device 105, and data storage device 106. Once the full-featured operating system is transferred and running in random access memory 102, other application executable files and data files may be accessed in data storage device 106. One such executable file is the dynamic non-volatile memory update program that contains most of the dynamic update processing logic of the present invention. The remaining portions of the processing logic of the present invention reside in the memory image itself. Files containing memory images and a file containing a predefined recovery BIOS image used for recovery mode updates also resides on data storage device 106. A memory image, of which a BIOS image is a specific type, is the binary contents of the target update area in non-volatile or flash memory. The dynamic update processing logic may be activated and executed using means well known to those of ordinary skill in the art. The dynamic update processing logic thus activated includes

logic for handling a normal update mode and the recovery update mode. The architecture and operation of the dynamic update processing logic is described below.

As indicated above, the preferred embodiment operates in two basic modes: a normal update mode and a recovery update mode. The present invention also includes hardware means for selecting one of these two available update modes. In the preferred embodiment, this means of selecting an update mode is implemented as a jumper on a circuit board in the computer system. The jumper is used to modify address line **16** in an interface to the flash memory device. By setting the jumper to a first setting, the normal BIOS map illustrated in FIG. **2** and the corresponding normal BIOS update utility may be used. When the jumper is set to a second setting, a recovery BIOS map is configured and the recovery update mode is enabled. Other means for selecting one of two modes in a computer system, such a switch or push button will be apparent to those skilled in the art.

Referring now to FIGS. **3***a* and **3***b*, the effect of the jumper in a recovery mode setting on the memory configuration in flash memory is illustrated. In FIG. **3***a*, a normal BIOS map **200** illustrates the configuration of flash memory while the jumper is set in a normal mode setting. This configuration is the same as that described above and illustrated in FIG. **2**. On power up or after a system reset, the processor jumps to a location in normal system BIOS **201** indicted by arrow **311** in FIG. **3***a*. The processing logic in normal system BIOS **201** then takes control and initializes the computer system for normal operations. If, however, normal system BIOS **201** is corrupted such as following an aborted BIOS update operation, the execution within normal BIOS **201** will be unpredictable. Thus, without a recovery mode feature, the computer system with a corrupted normal system BIOS **201** will be non-functional.

If the initiation of the normal system BIOS **201** is unsuccessful because of a corrupted BIOS **201**, the recovery mode may be selected by switching the selection means (i.e. jumper) to a recovery mode. In a recovery mode, address line **16**, used to address locations within flash memory, is modified to a complemented state. When so modified, the upper and lower halves of flash memory are logically flipped. Thus, address FFFFFH maps to location 8FFFFH and location F0000H maps to location 80000H. The remapped memory configuration in recovery mode is illustrated in the recovery BIOS map illustrated in FIG. **3***b*. In recovery mode, normal system BIOS **201** is remapped to a location **301** in the recovery BIOS map. Similarly, recovery BIOS block **202** is remapped to location **302** in the recovery BIOS map **300**. Memory areas **203**, **204**, and **205** are also remapped to locations **303**, **304** and **305** respectively.

Once the recovery mode is set using the jumper and a recovery BIOS map is achieved as illustrated in FIG. **3***b*, the processor of the computer system may then be power-up initialized or reset. Upon power up or reset, the processor jumps to a location within the protected recovery BIOS block **302** at a location indicated by arrow **312**. In this manner, the flash device memory map may be reconfigured in a recovery mode thereby activating recovery mode processing logic residing in recovery BIOS block **302**. Thus, using the mode selection means, either a normal system BIOS **201** or a recovery BIOS **302** may be activated. Once either the normal BIOS or recovery BIOS is activated, the dynamic BIOS update processing logic of the present invention may be retrieved from data storage device **106** and loaded into random access device **102** for execution by the processor **101**. A mode-indicating data item is also set for the dynamic update processing logic. This mode indication specifies whether the computer system is operating in a normal or a recovery mode.

Once activated, the dynamic update processing logic of the present invention operates in a manner described in detail below and illustrated in flow charts of FIGS. **5** and **6**. The dynamic BIOS update processing logic provides two sets of features depending on whether a normal mode or a recovery mode has been selected. In the normal mode, a robust human interface is provided. This interface provides graphical presentations for menu selection, file selection, help information, and status messages for updating any flash area including the system BIOS area. This graphical user interface package provides subroutines to define and display color user interface screens. This package also supports on line help to aid the user.

In a recovery mode, the recovery BIOS processing logic is designed only to bring critical portions of the system up without the advantage of configuration information. The recovery BIOS assumes that the normal system BIOS has been corrupted. Thus, the sole function of the recovery BIOS is to enable the system to a point where the normal system BIOS can be loaded into the flash memory device. The recovery BIOS performs automatically on power up without user intervention. Audible beep codes issued from a speaker of the computer system indicate the status of the programming of the BIOS area. The recovery BIOS is a subset of the normal BIOS.

When executing out of normal BIOS, the dynamic update program displays a menu of options for user selection. These options include: verification of a flash memory area, saving a selected flash memory area, updating a flash memory area, and exit. The dynamic update program then interfaces with the normal system BIOS to determine the areas in flash memory that are available for reading or writing. A submenu is then displayed indicating what areas are available for reading or writing. The user may then select the area from this menu for a verify, save, or update operation. Additional menus are displayed in order to prompt the user to enter file names for save/verify/update operations and for verifying file information before an update. Additional menus are also used to provide information to the user regarding a saved file image. The operation is then performed and the user is informed of its completion status. A normal update mode can be used to update either the normal system BIOS area **201** and **205**, the user reserved area **203**, or the system reserved area **204**. These areas can be updated either separately or in combination. Separately programmable area **203**, or other separately programmable areas, may contain LAN processing logic, SCSI processing logic, other network logic or data, video data or control logic, other hardware or software specific logic or data, or any other application specific processing logic in an area of flash memory. Areas **201** and **205** reside in the same physical block and thus are erased at the same time.

As mentioned above, the dynamic update program may be used to update any of the flash memory areas. Updating the normal system BIOS area was described above. Notable extensions of the present invention include updating the area **203**, or other separately programmable areas, with LAN processing logic, SCSI processing logic, video data or control logic, other hardware or software specific logic or data, or any other application specific processing logic in an area of flash memory. If a flash memory area containing LAN processing logic, SCSI processing logic, or network logic or data is updated, a computer system may be configured for a particular network connection. In prior art networked computer systems, a network boot PROM is neces-

sary for each specific network. It is necessary to change PROMs or add a different network circuit board in order to reconfigure a computer system for a particular network.

By using the present invention, flash memory areas other than the system BIOS area, such as area **203**, may be programmed with LAN processing logic, SCSI processing logic, other network logic or data, video data or control logic, other hardware or software specific logic or data, or any other application specific processing logic in an area of flash memory. The programming of these non-system BIOS update areas is accomplished using the same dynamic update utility procedures described for use in updating the system BIOS area. In addition, the system BIOS may be configured to scan the non-system BIOS update areas on system boot up in order to determine if the code residing there should be executed in order to boot the computer system in a different manner, such as from a network. Thus, the implementation of the dynamic update of a non-system BIOS flash memory area may involve an additional step of enabling the system BIOS scanning of the non-system BIOS update areas.

This enabling step may be accomplished in several ways. First, a hardware switch in the computer system or a particular keystroke sequence may be provided to activate the scanning of the update area following programming of the area. Another alternative embodiment of the enabling step is to update the system BIOS area subsequent to an update of non-system BIOS areas. By updating the system BIOS area, a data item may be set that enables the scanning of the non-system BIOS areas. In a third alternative embodiment, the system BIOS may be configured to always scan the non-system BIOS areas and execute the code there if a particular data pattern is detected at a fixed location within the non-system BIOS area. In any of these embodiments, a non-system BIOS area of flash memory may be dynamically updated thereby eliminating the need for network specific hardware or PROM devices. Using the updated non-system BIOS area, a computer system may be booted from a network or operated in a diskless configuration.

Referring now to FIG. **4**, the architecture of the dynamic update program of the present invention is illustrated. In the preferred embodiment, there are five separate compilation modules (**470**, **471**,**472**, **473**, **475**) that are linked together to form the dynamic update program **490** of the present invention. The main processing module **470** assumes all control, decision making, and interfaces with the remaining four modules. The memory image interface module **471** is used to process (i.e. read and write) memory images. A memory image is the binary contents of the target non-volatile memory area. Memory images are read from an external storage medium, such as a magnetic disk device, which contains a plurality of memory image files **479**. Memory images are written to files **478** in the external storage medium for permanent retention. The memory image files **479** each contain a header with various information on the memory image.

In order to update a non-volatile memory area, update logic must be used that is not residing in the non-volatile memory area being updated. If logic was used that resided in the non-volatile memory area being updated, the update operation could change or corrupt this logic. However, in order to achieve the objective of not implementing hardware specific procedures, it is advantageous to store the necessary update logic in each of the memory images **479** being updated. This update logic is stored in the memory images **479** in the form of protected software procedures. An important reason for storing the protected procedures in the memory image **479** itself is to partition hardware dependent code into the memory image **479** while maintaining hardware independent code in the dynamic update program **490**, which is executed from RAM. Thus, the same dynamic update program **490** can be used for many different hardware configurations, since the update logic in each memory image **479** handles hardware specific concerns. When a user requests an update operation, the dynamic update program **490** requests the normal system BIOS **481** to copy these procedures to a safe random access memory position specified by the dynamic update program **490**. This operation ensures that memory image itself defines and maintains these protected procedures and that they do not reside in the updated address space during flash updates. The protected update procedures module **472** is responsible for obtaining these protected procedures from the update memory image **479** and making them available to the main processing module **470** of the dynamic update program **490**.

The video interface module **473** performs all required interaction with the video display device **474**. This interaction includes displaying menus, displaying defined error messages, displaying status messages, and obtaining required information on memory image locations (i.e. file names). A user keyboard input means **480** is also provided for receiving user input and command selections. The flash interface module **475** provides a bridge between a set of unprotected BIOS procedures **476** and the main processing module **470**. Unprotected BIOS procedures **476** such as "Read Flash Memory" do not require the use of the protected procedures defined earlier. These unprotected BIOS procedures **476** are accessed through a processor interrupt (INT **15**) thereby providing access to the flash memory device **477**. The interface specifications for both the protected and unprotected procedures are provided below just before the Claims.

OPERATION OF THE PREFERRED EMBODIMENT

The processing logic of the dynamic update program of the preferred embodiment is operably disposed within random access memory **102** and executed by processor **101** of the computer system described above. The processing logic of the present invention may equivalently be disposed in other memory means accessible to processor **101** for execution. The processing logic of the dynamic update program **490** can be a separately compiled and loaded entity or incorporated as part of a larger software system. In either case, a means for activating the processing logic of the present invention may be performed using the techniques described above. Once activated, the processing logic of the present invention operates in the manner described below and illustrated in the flow charts of FIGS. **5** and **6**.

Referring now to FIG. **5**, the dynamic update program of the present invention starts at block **501** on activation. At decision block **502**, a test is performed to determine if the normal mode or recovery mode has been selected. As described earlier, the normal BIOS and the recovery BIOS both set a mode indicating data item when either BIOS is activated. By accessing this data item in decision block **502**, the active mode can be determined. If the recovery mode is selected, processing path **503** is taken to processing block **505** where a predefined normal system BIOS image is retrieved from the data storage device. Next, the recovery BIOS is accessed to retrieve the protected BIOS update procedures (processing block **506**). The protected procedures retrieved in block **506** include an erase BIOS proce-

dure and a program BIOS procedure. Next, the computer system is prepared for the erasure and programming of flash memory (processing block **535**). Preparing the computer system for erasure and programming of flash memory involves operations including disabling cache and shadowing functions, enabling write operations to flash, and other hardware specific operations. Once the protected procedures are retrieved and the computer system is prepared, the erase BIOS procedure is activated to erase the normal system BIOS area in flash memory. Once the erase operation is complete, the program BIOS procedure is used to load the predefined BIOS image into the normal system BIOS area of flash memory (programming block **507**). Once this operation is complete, the processing logic of the dynamic update program terminates at processing block **508**. The user is notified of completion by an audible beep code issued from a speaker of the computer system. At the termination of the recovery mode processing, the jumper or mode selection means can be switched back to a normal mode selection and the computer system can be restarted or power up initialized in order to transfer control to the newly loaded normal system BIOS in flash memory.

Referring again to decision block **502**, if a normal mode has been selected, processing path **504** is taken to processing block **510** where a main menu of normal update options are displayed to the user. These options include a verify flash area operation, a save flash area operation, an update flash area operation, and an exit operation. Once a menu has been presented to the user, a menu selection is retrieved from input device **104** in processing block **511**. If the exit operation is selected in decision block **512**, processing path **513** is taken to termination block **515** where execution of dynamic update program terminates.

If, however, the exit operation is not selected, processing path **514** is taken to processing block **516** where an unprotected normal system BIOS procedure is activated to retrieve active flash areas in BIOS that may be verified, saved, or updated. The active flash areas are displayed in a secondary menu in processing block **517**. The user is again prompted to enter a menu selection through input device **104** in processing block **518**. If a verify flash area is selected, processing path **520** is taken to the bubble labeled A illustrated in FIG. **6**. If a save flash area operation is selected, processing path **523** is taken to the bubble labeled B in FIG. **6**. Similarly, if an update flash area operation is selected, processing path **526** is taken to the bubble labeled C in FIG. **6**. If an inappropriate (i.e. Return to Main Menu) command selection is made, processing path **527** is taken to the bubble labeled D where control returns to processing block **510** where the main display menu is again presented to the user.

Referring now to FIG. **6**, the processing logic for each of the three flash memory area operations is depicted. If a verify flash memory operation is selected, the processing logic below the bubble labeled A is executed. In processing block **601**, the user is prompted to enter the name of a file that will be compared against the specified flash memory area. If a single file is not large enough to contain the entire contents of the specified flash memory area, several files containing portions of the comparison flash data image contents may be chained together using a file chaining technique well known in the art. In the loop between processing block **602** and decision block **604**, the contents of the comparison file or chained files are read and compared with the contents of the specified flash memory area. If differences are found, the loop terminates with a verify error. The verification process continues until the entire contents of the comparison file and associated chained files are compared with the contents of the selected flash memory area. When verification is complete, processing path **606** is taken to processing block **607** where the comparison results are displayed to the user. Processing then continues at the bubble labeled E illustrated in FIG. **5** where the operator is prompted for the next command selection.

Referring again to FIG. **6**, the processing logic associated with the save flash memory area command is illustrated below the bubble labeled B. When the user selects a save flash memory area command, processing control is transferred to processing block **608** where the user is prompted for the entry of a file name which will be used to store the selected flash memory area. Multiple files may be chained together using well known techniques if a single file is not large enough to store the entire selected flash image area. A loop is then initiated between processing block **609** and decision block **611** where the contents of the selected flash memory area is read (processing block **609**) and then written to the specified file or chained files (processing block **610**). The saving operation continues until the entire contents of the selected flash memory area have been transferred to the specified file or files. When this occurs, processing path **613** is taken to processing block **614** where the status of the save operation is displayed to the user. Processing then continues at the bubble labeled E illustrated in FIG. **5** where the operator is prompted for the entry of the next command selection.

Referring again to FIG. **6**, the processing logic for the update flash area command is illustrated below the bubble labeled C in FIG. **6**. When the update flash area command is selected by the user, processing control is transferred to processing block **615** where the user is prompted for the entry of a file name that contains a flash data image that will be transferred into the specified flash memory area. In order for the update utility of the present invention to handle various computer configurations and flash memory devices, the update processing logic is embedded in the normal system BIOS as protected procedures. The computer system is prepared for the erasure and programming of flash memory (processing block **635**). The flash memory area is erased in processing block **636**. These protected procedures are retrieved from the BIOS in processing block **616** for use by the dynamic update utility of the present invention. These protected update procedures are used by the dynamic update utility to transfer the contents of the specified file or files into the specified flash memory area (processing block **617**). This update operation continues until the entire contents of the specified file have been transferred to the specified flash memory area. When the update is complete (processing path **620**), the status results of the update operations are displayed to the user in processing block **621**. If the update requires a computer system reboot (processing path **623**), the newly updated BIOS may be activated by rebooting the computer system in processing block **625**. If no reboot is necessary, processing path **624** is taken to the bubble labeled E illustrated in FIG. **5** where the user is prompted for the entry of a new command selection.

Thus, a computer system wherein a portion of code/data stored in a non-volatile device that can be dynamically modified or updated without removing covers or parts from the computer system is described.

Although the present invention has been described herein with reference to a specific embodiment, many modifications and variations therein will readily occur to those skilled in the art. Accordingly, all such variations and modifications are included within the intended scope of the present invention as defined by the following claims.

BIOS FLASH MEMORY PROCEDURES

A. Unprotected BIOS Procedures

The following BIOS Service Calls are for the FLASH Memory Interface.

INTERRUPT = 15H

FUNCTION (AH) = 0DBH

SUBFUNCTIONS (AL)

00H Read FLASH Memory Area

Input:	DS =	Segment of output file header	
	SI =	Offset of output file header	
	ES =	Segment of buffer to place read info	
	DI =	Offset of buffer to place read info	
Output:	AH =	Status out	0 = Successful Read 1 = Invalid Input

01H Report FLASH Memory Areas

Input	CL =	Logical Area Number (0 – 0FFh)	
Output	DI =	offset of pointer to 32 byte area-info structure	
	ES =	segment of pointer to 32 byte area-info structure	
	AH =	Status out	0 = Successful 1 = Invalid Input

02H Get Protected Procedure Size

Input:	none	
Output:	BX =	Size of protected procedures in bytes

03H Prepare for FLASH Erasure

Input:	ES =	Segment of buffer to place protected procedures	
	DI =	Offset of buffer to place protected procedures	
	DS =	Segment of header of input file	
	SI =	Offset of header of input file	
	BX =	Segment of buffer for BIOS tables	
	DX =	Offset of buffer for BIOS tables	
Output	BX =	Amount of extra memory required by BIOS to complem operation	
	AH =	Status out	0 = Successful completion 1 = Invalid Input 2 = Invalid operation 3 = Invalid size 4 = Invalid data type

04H Is Recovery Mode Active

Input:	none		
Output:	BX	0 = Recovery mode active non-zero = Normal mode active	
	AH =	Status out	0 = Successful completion 1 = FLASH memory is not supported on this system

B. Protected BIOS Procedures

The protected procedures are accessed by a far call to the address returned in ES:DI by the "Prepare for Erasure" interrupt call and require the following interface:

AL = 00H Erase FLASH Memory Area

Input:	DS =	Segment of input file header	
	SI =	Offset of input file header	
	ES =	Segment of additional memory requested by BIOS	
	DI =	Offset of additional memory requested by BIOS	
	CH =	State	0 = Normal Mode 1 = Recovery Mode
	BX =	Segment of buffer for BIOS tables	
	DX =	Offset of buffer for BIOS tables	
Output:	AH =	Status out	0 = Successful Read 1 = Invalid Input 2 = Erase failure

AL = 01H Program FLASH Memory Area

Input:	CH =	State	0 = Normal Mode 1 = Recovery Mode
	DS =	Segment of input file header.	
	SI =	Offset of input file header	
	ES =	Segment of additional memory requested by BIOS	
	DI =	Offset of additional memory requested by BIOS	
	BX =	Segment of buffer for BIOS tables	
	DX =	Offset of buffer for BIOS tables	
Output:	AH =	Status out	0 = Successful Read 1 = Invalid Input 3 = Area not erased 4 = Verify Error

We claim:

1. A computer system comprising:

a processor for executing processing logic;

a programmable non-volatile semiconductor memory device coupled to said processor, said programmable non-volatile semiconductor memory device having BIOS firmware programmed therein, portions of said programmable non-volatile semiconductor memory device being write modifiable;

a random access memory (RAM) device coupled to said processor;

means for transferring a first portion of said BIOS firmware from said programmable non-volatile semiconductor memory device to said RAM, said first portion being specifically directed to updating other portions of said programmable non-volatile semiconductor memory device, said means for transferring coupled to said processor;

means for erasing a portion of said programmable non-volatile semiconductor memory device, said means for erasing coupled to said processor; and,

means for updating a second portion of said BIOS firmware in said programmable non-volatile semiconductor memory while said first portion of said BIOS firmware is executing from said RAM, said means for updating coupled to said processor.

2. The computer system as claimed in claim 1 wherein said programmable non-volatile semiconductor memory device is a flash semiconductor memory device.

3. The computer system as claimed in claim 1 wherein said programmable non-volatile semiconductor memory device is partitioned into at least two separately erasable and programmable partitions, a first partition storing recovery processing logic and data, a second partition storing normal operating system processing logic and data.

4. The computer system as claimed in claim 1 wherein said firmware is a basic input/output system (BIOS).

5. The computer system as claimed in claim 1 said means for updating further including:

means for selecting an update type, said update type being either a normal update type or a recovery update type, said means for selecting coupled to said processor.

6. The computer system as claimed in claim 3, wherein said means for updating further including:

means for selecting an update type, said update type being either a normal update type or a recovery update type, said means for selecting coupled to said processor;

normal update processing logic operably disposed in said random access memory;

means for activating said normal update processing logic if said normal update type is selected by said means for selecting; and

means for reprogramming said second partition if said normal update processing logic is activated.

7. The computer system as claimed in claim 6 said means for reprogramming further including:

protected update processing logic residing within said normal operating system processing logic stored in said second partition;

means for transferring said protected update processing logic to said random access memory; and

means for activating said protected update processing logic, said protected update processing logic controlling the programming of said second partition.

8. The computer system as claimed in claim 3 said means for updating further including:

means for selecting an update type, said update type being either a normal update type or a recovery update type, said means for selecting coupled to said processor;

means for activating said recovery processing logic if said recovery update type is selected by said means for selecting an update type; and

means for reprogramming said second partition if said recovery processing logic is activated.

9. The computer system as claimed in claim 3 wherein said first partition is electronically protected from erasure or modification.

10. A computer system comprising:

a processor for executing processing logic;

a programmable non-volatile semiconductor memory device coupled to said processor, said programmable non-volatile semiconductor memory device having BIOS firmware programmed therein, portions of said programmable non-volatile semiconductor memory device being write modifiable;

a random access memory (RAM) device coupled to said processor;

means for transferring a first portion of said BIOS firmware from said programmable non-volatile semiconductor memory device to said RAM, said first portion being specifically directed to updating other portions of said programmable non-volatile semiconductor memory device, said means for transferring coupled to said processor;

means for erasing a portion of said programmable non-volatile semiconductor memory device, said means for erasing coupled to said processor; and,

means for updating application specific BIOS firmware residing in a second portion of said programmable non-volatile semiconductor memory device while said first portion of said BIOS firmware is executing from said RAM, said means for updating coupled to said processor.

11. The computer system as claimed in claim 10 further including means for enabling scanning of said second portion of said programmable non-volatile semiconductor memory device containing application specific firmware.

12. The computer system as claimed in claim 10 wherein said application specific firmware includes LAN processing logic.

13. The computer system as claimed in claim 10 wherein said application specific firmware includes SCSI processing logic.

14. In a computer system having a processor for executing processing logic, a random access memory (RAM) coupled to said processor and a programmable non-volatile semiconductor memory device coupled to said processor, said programmable non-volatile semiconductor memory device having firmware programmed therein, portions of said programmable non-volatile semiconductor memory device being write modifiable, said firmware including a first portion programmed into an electronically write protected area of said programmable non-volatile semiconductor memory device, said first portion being specifically directed to updating said programmable non-volatile semiconductor memory device, a process for updating portions of said programmable non-volatile semiconductor memory device while said firmware is executing from said RAM, said process comprising the steps of:

transferring said first portion from said electronically protected area of said programmable non-volatile semiconductor memory device to said RAM;

executing said first portion to update a second portion of said programmable non-volatile semiconductor memory device; and

updating said second portion while said first portion is executing from said RAM.

15. The process as claimed in claim 14, said process further including the steps of:

selecting whether said first portion or said second portion is initially executed;

transferring said second portion from an unprotected area of said programmable non-volatile semiconductor memory device to said RAM, said step of transferring said second portion being performed if said second portion is selected for initial execution in said selecting step;

executing said second portion from said RAM if said second portion is selected for initial execution in said selecting step; and

executing said first portion from said RAM if said first portion is selected for initial execution in said selecting step.

16. The process of claim 15, wherein said second portion of said programmable non-volatile semiconductor memory device includes a second version of operating system firmware.

17. The process of claim 15, wherein said second portion of said programmable non-volatile semiconductor memory device includes application specific processing logic.

18. In a computer system having a processor for executing processing logic, a random access memory (RAM) coupled to said processor and a programmable non-volatile semiconductor memory device coupled to said processor, said programmable non-volatile semiconductor memory device having operating system firmware programmed therein, portions of said programmable non-volatile semiconductor memory device being write modifiable, a process for updating application specific firmware residing in a portion of said programmable non-volatile semiconductor memory device while said operating system firmware is executing, said process comprising the steps of:

transferring a first portion of said operating system firmware from said programmable non-volatile semiconductor memory device to said RAM, said first portion being specifically directed to reprogramming portions of said programmable non-volatile semiconductor memory device; and

executing said first portion of said operating system firmware to reprogram a second portion of said programmable non-volatile semiconductor memory device with application specific firmware.

19. The process as claimed in claim 18 further including a step of enabling scanning of said second portion of said programmable non-volatile semiconductor memory device containing application specific firmware.

20. The process as claimed in claim 18 wherein said application specific firmware includes LAN processing logic.

21. The process as claimed in claim 18 wherein said application specific firmware includes SCSI processing logic.

22. A computer system comprising:

a processor for executing logic instructions;

a programmable flash memory electrically connected to said processor, said programmable flash memory subdivided into a first part and a second part;

said first part of said programmable flash memory stores BIOS instructions for a normal update mode, wherein said normal update mode includes the instructions needed to save, verify, or update areas of flash memory including said second part of said programmable flash memory;

said second part of said programmable flash memory stores BIOS instructions for a recovery update mode, wherein said recovery update mode includes instructions needed to start the computer system and update the first part of said programmable flash memory.

23. The computer system described in claim 22, wherein said computer system includes a hardware device for selecting either said normal update mode or said recovery update mode.

24. The computer system of claim 23 wherein said hardware device is a jumper cable.

25. A method of updating a BIOS stored in a programmable non-volatile semiconductor memory device in a computer, said method comprising the steps of:

creating two portions of BIOS, a normal update portion and a recovery update portion and storing each portion in a section of said programmable non-volatile semiconductor memory device;

using a hardware device to switch between two start-up modes in said computer;

choosing to load into RAM memory either said normal update portion of BIOS or said recovery update portion of BIOS depending on setting of said hardware device;

loading the chosen portion of BIOS into said RAM memory; and

updating the portion of BIOS not loaded into said RAM memory.

* * * * *

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☑ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.